世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

TURING 图灵电子与电气工程丛书

ELSEVIER 爱思唯尔

tyw藏书

PIC项目实战

Advanced PIC Microcontroller Projects in C

From USB to RTOS with the PIC18F Series

[塞浦] Dogan Ibrahim 著
李中华 张雨浓 郐依林 等译



人民邮电出版社
POSTS & TELECOM PRESS

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源



图灵电子与电气工程丛书

PIC项目实战



Advanced PIC Microcontroller Projects in C

From USB to RTOS with the PIC18F Series

[塞浦] Dogan Ibrahim 著
李中华 张雨浓 邬伊林 等译

Newnes

INCLUDES FREE NEWNES ONLINE MEMBERSHIP

人民邮电出版社
北 京

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com
tyw藏书

图书在版编目（CIP）数据

PIC项目实战 /（塞浦）伊瓦海姆（Ibrahim, D.）著；李中华等译. —— 北京：人民邮电出版社，2010.7
（图灵电子与电气工程丛书）
书名原文：Advanced PIC Microcontroller Projects in C:From USB to RTOS with the PIC18F Series
ISBN 978-7-115-22917-5

Ⅰ. ①P… Ⅱ. ①伊… ②李… Ⅲ. ①单片微型计算机 Ⅳ. ①TP368.1

中国版本图书馆CIP数据核字（2010）第078190号

内 容 提 要

本书是一本关于在PIC18F微控制器上用C语言进行项目编程的经典之作。全书共10章，深入介绍了PIC18F系列微控制器和mikroC编译器的特性，并结合20多个完整可行的项目实例，阐述了使用mikroC语言设计PIC微控制器应用的方法，以及SD卡、USB总线、实时操作系统等的原理与应用。

本书适用于大学高年级学生和工程技术人员，以及PIC18F系列微控制器编程与应用的爱好者。

图灵电子与电气工程丛书

PIC项目实战

◆ 著　　[塞浦] Dogan Ibrahim
译　　李中华　张雨浓　邬依林　等
责任编辑　朱　巍
执行编辑　罗词亮

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市潮河印业有限公司印刷

◆ 开本：787×1092　1/16
印张：22
字数：606千字　　2010年7月第1版
印数：1－3 000册　　2010年7月河北第1次印刷

著作权合同登记号　图字：01-2009-5729号

ISBN 978-7-115-22917-5

定价：55.00元

读者服务热线：(010)51095186　印装质量热线：(010)67129223
反盗版热线：(010)67171154

PDG

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

前　　言

微控制器是集数据存储器、程序存储器、串行和并行I/O、定时器以及内部中断和外部中断于一体的微处理器系统。这样的一枚集成芯片售价仅2美元甚至更低。约40%的微控制器应用于PC、激光打印机、传真机、智能电话等办公设备，约1/3的微控制器应用于CD播放机、高保真设备、视频游戏机、洗衣机和炊具等消费类电子产品，其余的微控制器应用于通信、汽车以及军事领域。

本书是专门为大学高年级学生、工程技术人员以及PIC18F系列微控制器编程与应用的爱好者而编写的。本书假定读者已经修完数字逻辑设计课程，并且至少能使用一门高级编程语言编写程序。掌握C语言且熟悉至少一款PIC16F系列微控制器，将更有利于学习本书。本书不要求掌握读者具备汇编语言程序的知识，因为本书所有的项目都是以C语言为基础的。

第1章介绍了微控制器的基本特点，讨论了计数系统，描述了数制之间的转换。

第2章回顾了PIC18F系列微控制器，详细描述了该系列微控制器的各种特性。

第3章简要介绍了C语言的基础知识，剖析了mikroC编译器的特性。

第4章阐述了mikroC语言的高级特性，并以实例的形式讨论了内置函数及函数库。

第5章探讨了PIC18F系列微控制器的各种软硬件开发工具，并以实例的形式讨论了各种商业应用开发套件和诸如模拟器、仿真器、内部电路调试器等开发工具。

第6章提供了部分使用PIC18F系列微控制器和mikroC编译器的简单项目。所有的项目都是基于PIC18F452微控制器的，并且全部通过测试。该章对于那些学习PIC微控制器的新手以及想掌握如何使用mikroC语言设计PIC18F微控制器应用的读者都是很有帮助的。

第7章介绍了如何在PIC18F微控制器设计中使用SD存储卡。SD存储卡的原理介绍将以实际项目例子来展开。

第8章回顾了非常流行的USB总线，并通过实际项目讨论了这种总线的基本原理，这些项目阐明了如何设计通过USB总线同PC通信的基于PIC18F的项目。

当前，CAN总线广泛应用于汽车电子。第9章简要介绍了CAN总线的原理，讨论了如何使用CAN总线接口设计基于PIC18F微控制器的项目。

第10章介绍了实时操作系统（RTOS）和多任务。该章不仅给出了RTOS系统的基本原理，还提供了简单的多任务应用程序。

本书附属资源[①]包括本书中所有项目的程序源文件和十六进制文件，此外，还附有mikroC编译器软件（大小为2KB的受限版本）。

Dogan Ibrahim
2007年于伦敦



① 请登录图灵公司网站（www.turingbook.com）本书页面免费注册下载。——编者注

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com
tyw藏书

目　　录

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书



世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

第 1 章　微型计算机系统

1.1　引言

微型计算机通常是指至少包括有微处理器、程序存储器、数据存储器、I/O设备的系统。有些微型计算机系统还包括有定时器、计数器、模数转换器等部件。因此，微型计算机系统可以是包括有硬盘、软盘和打印机的大的计算机，也可以是集成在一块芯片上的嵌入式控制器。

本书将只考虑由单片机组成的微型计算机。这样的微型计算机系统又被称作微控制器(microcontroller)，它们被广泛应用于家电产品，如微波炉、电视遥控器、烹饪电器、高保真设备、CD播放器、个人电脑和电冰箱等。市面上的微控制器种类繁多，让人眼花缭乱。本书将围绕Microchip公司制造的PIC（可编程接口控制器）系列微控制器的编程和系统设计展开讨论和介绍。

1.2　微控制器系统

微控制器就是单芯片的计算机。术语微（micro）意指设备很小，术语控制器（controller）意指用作控制应用。微控制器的另一表述是嵌入式控制器（embedded controller），因为大多数控制器都被嵌入在所控制的设备中。

微处理器(microprocessor)与微控制器在很多方面都是不同的。主要区别是微处理器的运行需要外接其他设备，如程序存储器、数据存储器、I/O设备、外部时钟电路；而微控制器却将所有支持芯片整合在单一芯片中。所有微控制器都执行存储在程序存储器中的指令集或者用户程序。微控制器逐一地从程序存储器中取出指令并解码，然后再执行所要求的操作。

微控制器传统上是使用目标设备的汇编语言来编程的。虽然汇编语言的指令速度较快，但存有不少缺点。首先，汇编程序是由助记符构成的，这使得学习和维护汇编语言程序相当困难。此外，由不同厂商制造的微控制器有着不同的汇编语言，因此当用户使用新的微控制器时都必须学习新的汇编语言。

微控制器也可以使用诸如BASIC、PASCAL或C等高级语言来编程。高级语言学习起来比汇编语言容易得多，这使得大型复杂程序的开发更为简便。本书将学习如何使用mikroElektronika公司开发的流行的C语言mikroC来对PIC微控制器进行编程。

从理论上讲，一块单独的芯片就足以运行微控制器系统。然而，在实际的应用中，微型计算机需要额外的组件来同外部的环境进行交互。随着PIC系列微控制器的出现，电子项目的开发在短短几个小时之内就可以完成。

微型计算机基本上都执行存储在程序存储器中的用户程序。在程序的控制下，微型计算机从外部设备接收数据（输入），然后执行操作，最后再把数据发送到外部设备（输出）。举个例子，在一个基于微控制器的炉温控制系统中，微型计算机通过温度传感器读入温度值，然后控

PDG

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

tyw藏书

制加热器或者风扇将温度保持在期望值。图1-1给出了一个简单的温度控制系统的方框图。

图1-1所示的系统非常简单。复杂一点儿的系统还可能包括设置温度值的键盘和显示温度值的LCD（液晶显示器）。图1-2给出了一个更复杂的温度控制器系统的方框图。

2

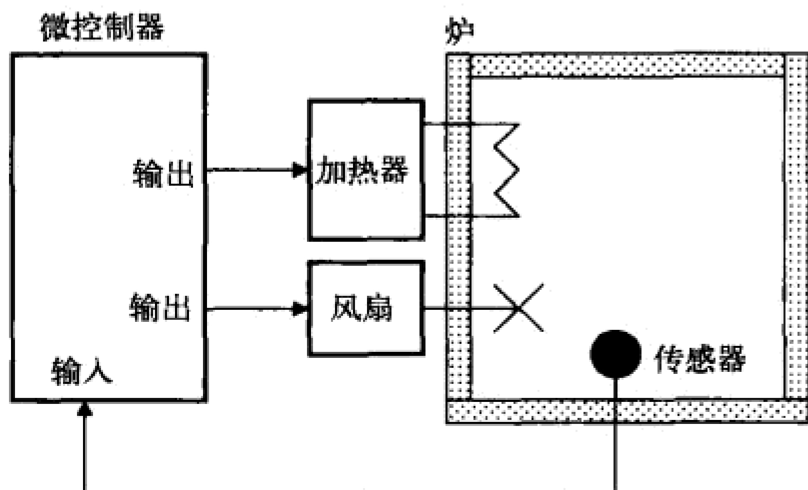


图1-1　基于微控制器的炉温控制系统

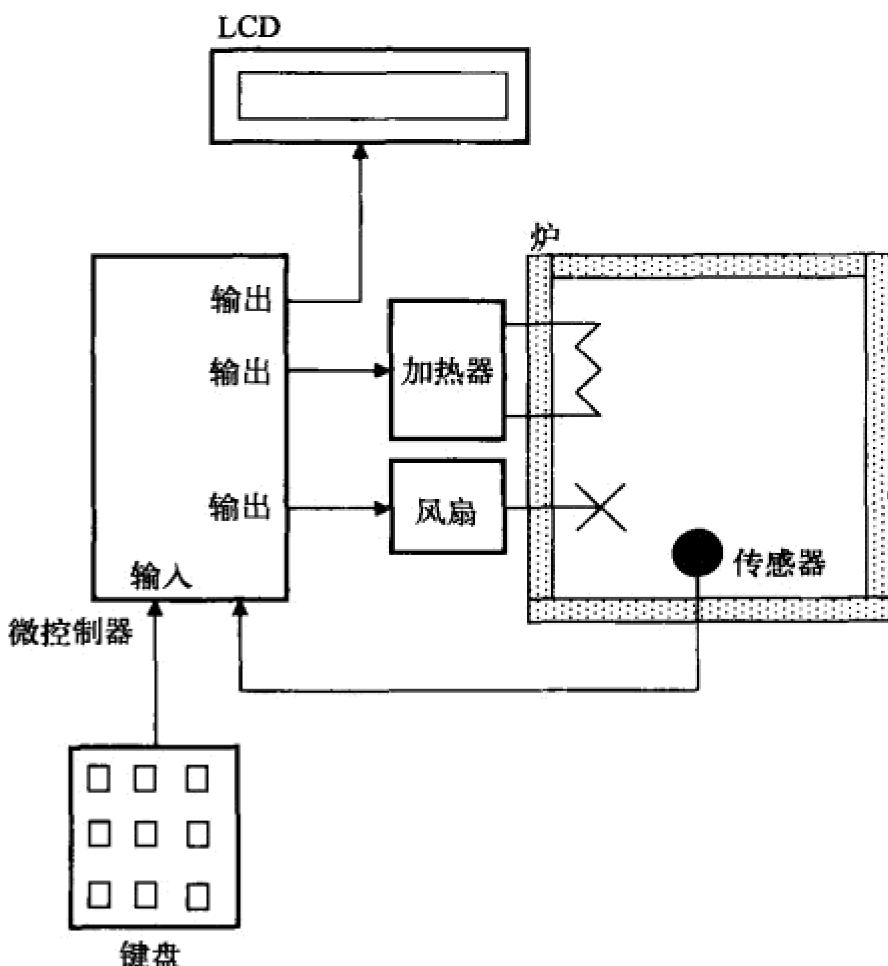


图1-2　带键盘和LCD的温度控制系统

3

可以将控制系统设计得更复杂一些（如图1-3所示）。如增设一个报警器，当温度超出期望的范围时，触发报警器。同时，将每秒的读取的温度值发送给PC存档并进一步处理。例如，可以在PC上绘制每天的温度曲线图。正如读者可以看到的，因为微控制器是可编程的，所以最终系统的简单或者复杂由设计人员来决定。

微控制器是非常强大的工具，它允许设计人员在程序的控制下构建复杂的I/O数据操作。微控制器可按照所处理数据的总线宽度来进行分类。8位微控制器是最流行的，大多数基于微控制器的应用都采用这种微控制器。16位微控制器和32位微控制器的功能更加强大，但比较昂贵，在中小型一般用途的微控制器应用中通常不采用。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

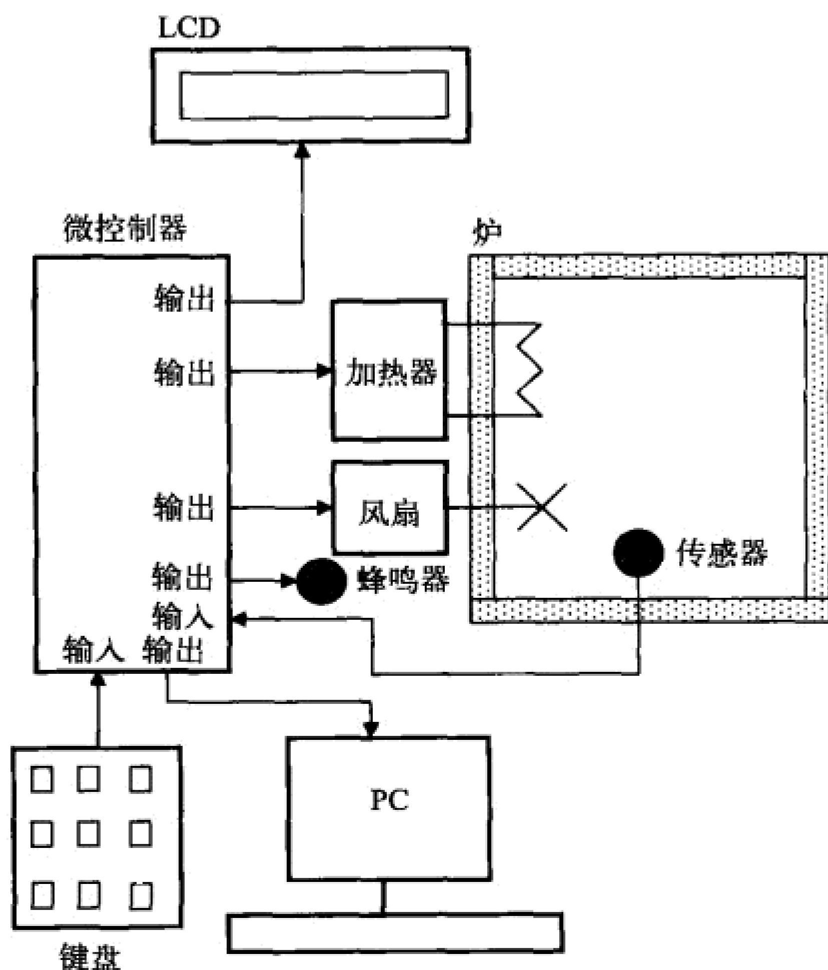


图1-3 更复杂的温度控制器

4

最简单的微控制器架构由微处理器、存储器和I/O设备组成。微处理器由中央处理单元（CPU）和控制单元（CU）组成。CPU好比微控制器的大脑，所有的算术运算和逻辑运算都在此进行。CU控制微处理器的内部运算，并向微控制器的其他部分发送信号以执行期望的指令。

存储器是微控制器系统的一个重要组成部分，可分为两类：程序存储器和数据存储器。程序存储器用来存储程序员编写的程序，通常是非易失性的（即断电后数据并不会丢失）。数据存储器用来存储程序运行中用到的临时数据，通常是易失性的（即断电后数据将会丢失）。

下面简要介绍6种基本的存储器。

1.2.1 RAM

RAM（随机存取存储器）是一种通用存储器，通常用来存储程序运行中的用户数据。RAM存储器是易失性的，因为在断电后它不能保留数据（即断电后数据将会丢失）。大多数微控制器都带有一定数量的内部RAM，通常都有256 B的空间，但某些微控制器的内部RAM空间可能会更大些也可能小些。例如，PIC18F452微控制器带有1 536 B的内部RAM。存储器通常可通过增加外部存储芯片来进行扩展。

1.2.2 ROM

ROM（只读存储器）通常用来存储程序或固定用户数据。ROM是非易失性的。如果将ROM断电然后再上电，最初的数据仍然保存在ROM中。ROM是在制造过程中完成编程操作的，用户不能改变它的内容。如果设计人员开发了一个程序并且希望复制成千上万份时，ROM存储器就非常有用。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

1.2.3 PROM

PROM（可编程的只读存储器）是一种可编程的ROM，最终用户可使用一种叫做PROM编程器的设备对其进行编程。一旦PROM被编程，它的内容将不能更改。通常在只需要少量存储空间的低产量应用中使用PROM存储器。

5

1.2.4 EPROM

EPROM（可擦除的可编程只读存储器）与ROM类似，但是EPROM可以使用合适的编程设备进行编程。EPROM存储器的芯片顶部有一个小小的透明玻璃窗，在玻璃窗上使用强烈的紫外线可以擦除EPROM中的数据。在EPROM存储器被编程以后，应该使用黑色胶带覆盖玻璃窗以防止意外地擦除数据。EPROM在重新编程之前必须擦除数据。许多高档的微控制器都内置有EPROM，用来存储用户程序。可以对这些存储器进行擦除和重新编程，直到用户满意为止。有些版本的EPROM（如OTP，一次性编程）允许使用合适的编程器设备对它进行编程，但是不能进行擦除。OTP存储器比EPROM要便宜很多。当一个项目已被完全开发出来并且需要进行大量的复制时，OTP是很有用的。

1.2.5 EEPROM

EEPROM，即电可擦除只读存储器，是一种非易失性的存储器，可以使用合适的编程器设备擦除存储内容并重新编程。EEPROM常用来存储配置信息、最大值和最小值、身份数据等。有些微控制器带有内置的EEPROM。例如，PIC18F452有一个256 B的EEPROM，每一字节数据都可以通过应用软件直接进行擦除和编程。EEPROM的访问速度通常很慢。EEPROM芯片比EPROM要贵得多。

1.2.6 Flash EEPROM

Flash EEPROM是EEPROM的另一种版本，在微控制器运用中非常流行，通常用来存储用户数据。Flash EEPROM是非易失性的而且速度很快。其存储的数据可以通过合适的编程器设备进行擦除然后重新编程。有些微控制器只有1 KB的Flash EEPROM，而有些则有32 KB甚至更大。PIC18F452微控制器就有32 KB的Flash EEPROM。

1.3 微控制器的特点

不同制造商生产的微控制器的结构和性能都各不相同。有些微控制器适合某些特殊的应用，而另外一些控制器则可能完全不适合同样的应用。本章将介绍大多数微控制器所共有的特点。

6

1.3.1 工作电压

大多数微型控制器都在标准的 + 5 V逻辑电压下工作。有些微控制器只需 + 2.7 V就可以工作，而有些微控制器能毫发无损地承受 + 6 V的电压。制造商会在数据表上标注出微控制器允许的工作电压的信息。PIC18F452微控制器能在 + 2 V～ + 5.5 V的电压范围内运行。

通常，当微控制器采用交流电源适配器或干电池供电时，电压调节器用来获得所需的工作电压。例如，如果微控制器要使用9 V电池来获得5 V的工作电压，则需要选用一个5 V的调节器。

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

1.3.2 时钟

所有的微控制器都需要时钟（或振荡器）来运行，通常是由连接到微控制器的外部定时设备提供。大多情况下，这些外部定时设备由一个晶体和两个小电容构成。有时候，它们由共振器或一对外部的电容电阻构成。有些微控制器带有内置的定时电路，而不需要外部的定时设备。如果应用对时间不敏感，那么使用外部或内部（如果可用的话）电阻电容定时部件是最好的选择，因为它们既简单又便宜。

指令的执行包括从存储器中读取指令并对其进行解码。这通常要占用好几个时钟周期，即指令周期。在PIC微控制器中，一个指令周期将耗时4个时钟周期。因而，微控制器运行的时钟频率是实际振荡器频率的1/4。PIC18F系列微控制器支持的时钟频率最高能达到40 MHz。

1.3.3 定时器

定时器是微控制器的重要组成部分。定时器实际上是由外部的时钟脉冲或微控制器的内部振荡器驱动的计数器。定时器可以是8位或16位宽度的。在程序控制下可以将数据装载入定时器，定时器的停止或启动由程序控制。大多数定时器都可以配置成：当它们达到某个数值时（通常是计数溢出时）产生中断信号。用户程序可以使用中断来执行微控制器内部同时间有关的精确运算。PIC18F 系列的微控制器都带有至少3个定时器。例如，PIC18F452微控制器就带有3个内置的定时器。

7

有些微控制器还提供捕获和比较设备，当外部事件发生时读取定时器的值，或者将定时器的值与预设的值进行比较，当达到这个值时就产生中断信号。大多数PIC18F微控制器都带有至少两个捕获和比较模块。

1.3.4 看门狗

大多数微控制器都至少有一个看门狗设备。看门狗实际上是由用户程序不断刷新的定时器。只要程序刷新看门狗失败，就会发生复位。看门狗定时器是用来检测系统问题的，如程序陷入了无休止的循环。这种安全特性能防止软件失控和避免微控制器执行无意义或不必要的程序代码。看门狗设备通常用于实时系统，有规律地检查一个或者多个活动的成功终结。

1.3.5 复位输入

复位输入用来在外部复位微控制器。复位是将微控制器拉回到一个已知状态，使程序从程序存储器内的0地址开始执行。外部的复位动作通常是将按键开关连接到复位输入引脚来实现的。当按下开关时，微控制器就会被复位。

1.3.6 中断

中断是微控制器中的一个重要概念。中断使得微控制器能快速地响应外部和内部（如定时器）事件。当中断出现时，微控制器就会离开正常的程序执行流程而跳转到一个特殊的程序段，即ISR（中断服务程序）。当ISR内的程序代码执行完成后，微控制器又会从ISR返回并继续执行正常的程序流程。

8

ISR开始于程序存储器的一个固定地址（有时又称为中断向量地址）。有些具有多个中断特性的微控制器只有一个中断向量地址，而另一些微控制器则拥有单独的中断向量地址，每个终端特性对应着一个中断源。中断是可以嵌套的，而一个新的中断可以暂停另一个中断的执行。

PDG

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

多中断能力的另一个重要特点就是，不同的中断源可分配不同的优先级。例如，PIC18F系列的微控制器具有低优先级和高优先级两个中断级别。

1.3.7　掉电检测器

掉电检测器（brown-out detector）很多微控制器中都很常用。如果供电电压低于标称值，则掉电检测器会复位微控制器。掉电检测器的这种安全特性常被用来防止微控制器在低电压下进行不可预知的操作，尤其是用来保护EEPROM类型存储器的存储信息。

1.3.8　模数转换器

模数转换器（A/D转换器）用来把模拟信号（如电压）转化成微控制器可以读取和处理的数字信号。有些微控制器内置有A/D转换器，也可以将外部的A/D转换器连接到任何类型的微控制器上。A/D转换器通常是8～10位，具有256～1 024的量化水平。大多数带有A/D转换功能的PIC微控制器都有多个A/D转换器，用以提供多路模拟输入通道。例如，PIC18F452微控制器带有10位8通道的A/D转换器。

A/D转换过程必须由用户程序启动，并耗时几百微秒才能完成。当转换完成时，A/D转换器通常会产生中断，从而使用户程序能很快读入所转换的数据。

A/D转换器在控制和监控应用中特别有用，因为大多数传感器（如温度传感器、压力传感器、力传感器等）输出的都是模拟电压。

1.3.9　串行输入/输出

9

串行通信（又称作RS232通信）允许微控制器通过串行电缆连接到其他的微控制器或PC。有些微控制器内置有USART（通用同步异步收发器）来实现串行通信接口。用户程序通常可以选择波特率（baud rate）和数据格式。如果没有提供串行输入/输出硬件，那么可以很容易开发软件通过微控制器的I/O引脚来实现串行数据通信。PIC18F系列微控制器都内置了USART模块。第6章将介绍在有USART模块和无USART模块的条件下如何编写mikroC程序来实现串行通信。

有些微控制器（如PIC18F系列）内置有SPI（串行外围接口）或I^2C（集成连接）硬件总线接口。这些接口使微控制器可以很容易地与其他设备兼容。

1.3.10　EEPROM数据存储器

EEPROM类型的数据存储器普遍应用于许多微控制器。EEPROM存储器的优点是程序员可以存储非易失性的数据并可以随时更改数据。例如，在一个温度监控应用中，可以将读得的最大温度值和最小温度值存储在EEPROM存储器中。如果因为某种原因断电，EEPROM存储器中的最后一次读数仍是可用的。PIC18F452微控制器带有256 B的EEPROM存储器。其他PIC18F系列的微控制器有更大的EEPROM存储器（如PIC18F6680带有1 024B）。mikroC语言提供了专门的指令来读写PIC微控制器的EEPROM存储器 。

1.3.11　LCD驱动器

LCD驱动器允许微控制器直接连接到外部的LCD显示器。这些驱动器并不常见，因为它们提供的大多数功能都可以通过软件实现。例如，PIC18F6490微控制器就内置有LCD驱动器模块。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

1.3.12 模拟比较器

当需要比较两个模拟电压的时候，就会用到模拟比较器。尽管大多数高端微控制器都应用了这种电路，但是在其他微控制器中它并不常见。PIC18F系列的微控制器内置有模拟比较器模块。 10

1.3.13 实时时钟

实时时钟使得微控制器能连续地接收绝对的日期和时间信息。大多数微控制器中并没有内置的实时时钟，因为同样的功能很容易通过一个专用的实时时钟芯片或是一个专门为此编写的程序来实现。

1.3.14 睡眠模式

有些微控制器（如PIC）提供内置的睡眠模式，它能执行指令停止内部振荡器，将功耗降低到一个非常低的水平。睡眠模式的主要目的是当微控制器空闲时节省电池的电量。通常可通过外部复位信号或者看门狗超时信号将微控制器从睡眠模式中唤醒过来。

1.3.15 上电复位

有些微控制器（如PIC）带有内置的上电复位电路，它能保持微控制器处于复位状态，直到所有的内部电路初始化完成。这个特性是非常有用的，它使得微控制器在启动时处于一个已知状态。也可以提供外部复位信号，当按下外部按钮开关时就能复位微控制器。

1.3.16 低功耗运行

低功耗运行在便携式应用中尤其重要，因为基于微控制器的设备都采用电池供电。有些微控制器（如PIC）在5 V电源电压下的工作电流低于2 mA，在3 V电源电压下仅有约15 μA的工作电流。其他的微控制器，特别是带有几个芯片的基于微处理器的系统，通常要消耗几百毫安甚至更大的电流。

1.3.17 电流拉出/灌入能力

如果将微控制器连接到需要大量工作电流的外部设备上，那么电流拉出/灌入能力是非常重要的。PIC微控制器能从每一个输出端口引脚灌入和拉出25 mA的电流。该电流通常足以驱动LED、小灯泡、蜂鸣器、小继电器等设备。将外部晶体管开关电路或继电器连接到输出端口引脚，可以增强电流的驱动能力。 11

1.3.18 USB接口

USB是当前非常流行的电脑接口规范，用来将各种外围设备连接到电脑和微控制器上。有些PIC微控制器（如PIC18F2x50）提供了内置的USB模块。

1.3.19 电机控制接口

有些PIC微控制器（如PIC18F2x31）提供了电机控制接口功能。

1.3.20 CAN接口

CAN总线是非常流行的总线系统，主要应用于自动化应用。有些PIC18F系列微控制器（如

PDG

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

tyw藏书

PIC18F4680）提供了CAN接口功能。

1.3.21　以太网接口

有些微控制器（如PIC18F97J60）提供了以太网接口功能，因而很容易用于基于网络的应用。

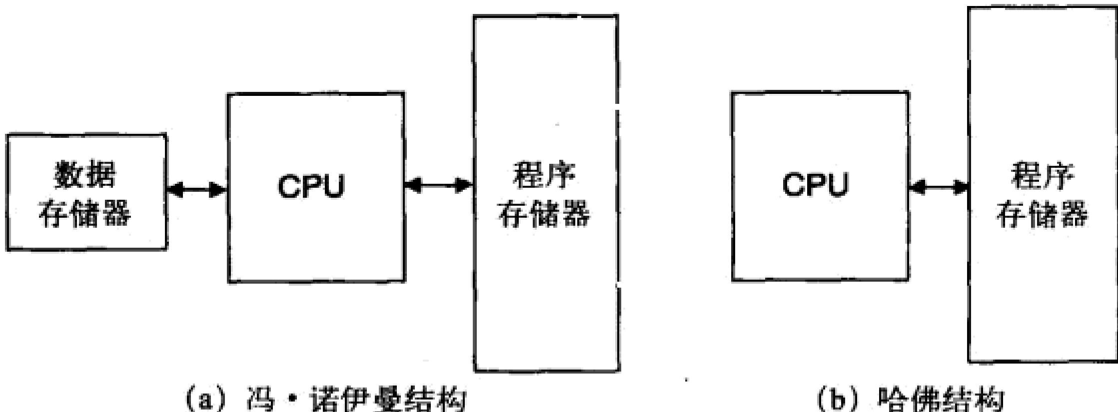
1.3.22　ZigBee 接口

ZigBee是一种类似于蓝牙的接口，主要用于低成本的无线家电。有些PIC18F系列微控制器提供了ZigBee接口功能，使得无线系统的设计非常简单。

1.4　微控制结构

微控制器通常有两种结构（如图1-4所示）。一种是大多数微控制器采用的冯·诺伊曼结构，所有的存储空间共享同一条总线，指令和数据都用这条总线。另外一种是PIC微控制器使用的哈佛结构，指令代码和数据分别使用独立的总线，允许同时读取代码和数据，从而达到改进性能的效果。

12



（a）冯·诺伊曼结构　　（b）哈佛结构

图1-4　冯·诺伊曼结构和哈佛结构

RISC 和 CISC

RSIC（精简指令集计算机）和CISC（复杂指令集计算机）都是微控制器的指令集。在8位的RISC微控制器中，数据是8位宽的，但是指令字的宽度可以大于8位（通常是12、14或16位），指令在程序存储器中占一个字。因而指令可以在一个周期中完成读取和执行，从而改进了性能。

在CISC微控制器中，数据和指令都是8位宽的。CSIC微控制器通常有两百多个指令。数据和代码共享同一条总线，因而不能被同时读取。

1.5　数制

为了高效地使用微处理器或微控制器，需要掌握二进制、十进制和十六进制数制的知识。本节将为那些不熟悉数制及其相互转换的读者介绍数制的相关背景信息。

数制通常按照它们的基数来分类。日常生活中的数制使用10为基数，即十进制数系统。微处理器和微控制器通常使用以16为基数的数制，即十六进制。我们也使用以2为基数（即二进制）和以8为基数（即八进制）的数制。

13

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

1.5.1 十进制数系统

十进制数系统的数字包括0、1、2、3、4、5、6、7、8、9。下标10表示这是一个十进制形式的数。例如，十进制数235可以写成235_{10}。

通常，一个十进制数可以表示成如下的形式：

$$a_n \times 10^n + a_{n-1} \times 10^{n-1} + a_{n-2} \times 10^{n-2} + \cdots + a_0 \times 10^0$$

例如，十进制数825_{10}可以表示成：

$$825_{10} = 8 \times 10^2 + 2 \times 10^1 + 5 \times 10^0$$

类似地，十进制数26_{10}也可以表示成：

$$26_{10} = 2 \times 10^1 + 6 \times 10^0$$

或

$$3359_{10} = 3 \times 10^3 + 3 \times 10^2 + 5 \times 10^1 + 9 \times 10^0$$

1.5.2 二进制数系统

二进制数系统由两个数构成：0和1。下标2表示这是一个二进制形式的数。例如，二进制数1011又可以写成1011_2。

通常，一个二进制数可以表示成如下的形式：

$$a_n \times 2^n + a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} + \cdots + a_0 \times 2^0$$

例如，二进制数1110_2可以写成：

$$1110_2 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

[14]

类似地，二进制数1001110_2可以表示成：

$$1001110_2 = 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

1.5.3 八进制数系统

在八进制数系统中，有效的数字是0、1、2、3、4、5、6、7。下标8表示这是一个八进制形式的数。例如，八进制数23可以写成23_8。

通常，一个八进制数可以表示成如下的形式：

$$a_n \times 8^n + a_{n-1} \times 8^{n-1} + a_{n-2} \times 8^{n-2} + \cdots + a_0 \times 8^0$$

例如，八进制数237_8可以表示成：

$$237_8 = 2 \times 8^2 + 3 \times 8^1 + 7 \times 8^0$$

类似地，八进制数1777_8可以表示成：

$$1777_8 = 1 \times 8^3 + 7 \times 8^2 + 7 \times 8^1 + 7 \times 8^0$$

1.5.4 十六进制数系统

在十六进制数系统中，有效的数字是0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F。下标16或H表示这是一个十六进制形式的数。例如，十六进制数1F可以写成$1F_{16}$或$1F_H$。

通常，一个十六进制数可以表示成如下的形式：

$$a_n \times 16^n + a_{n-1} \times 16^{n-1} + a_{n-2} \times 16^{n-2} + \cdots + a_0 \times 16^0$$

例如，十六进制数$2AC_{16}$可以表示成：

$$2AC_{16} = 2 \times 16^2 + 10 \times 16^1 + 12 \times 16^0$$

类似地，十六进制数$3FFE_{16}$可以表示成：

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

15

$$3FFE_{16} = 3 \times 16^3 + 15 \times 16^2 + 15 \times 16^1 + 14 \times 16^0$$

tyw藏书

1.6 二进制数转换为十进制数

为了把二进制数转换为十进制数，要计算出2的幂的总和。

例1.1 将二进制数1011_2转换为十进制数。

解 计算出2的幂的总和：

$$\begin{aligned}1011_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 8 + 0 + 2 + 1 \\ &= 11\end{aligned}$$

即$1011_2 = 11_{10}$。

例1.2 把二进制数11001110_2转换为十进制数。

解 计算出2的幂的总和：

$$\begin{aligned}11001110_2 &= 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 128 + 64 + 0 + 0 + 8 + 4 + 2 + 0 \\ &= 206\end{aligned}$$

即$11001110_2 = 206_{10}$。

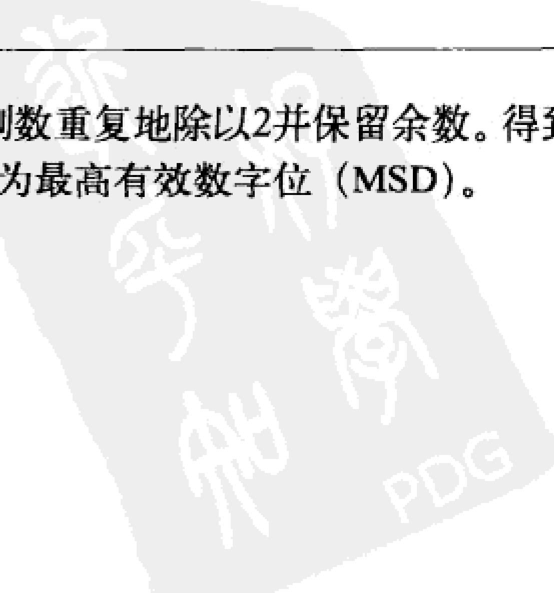
表1-1给出了二进制数对应的十进制数（0～31）。

表1-1 二进制数对应的十进制数

二进制	十进制	二进制	十进制
00000000	0	00010000	16
00000001	1	00010001	17
00000010	2	00010010	18
00000011	3	00010011	19
00000100	4	00010100	20
00000101	5	00010101	21
00000110	6	00010110	22
00000111	7	00010111	23
00001000	8	00011000	24
00001001	9	00011001	25
00001010	10	00011010	26
00001011	11	00011011	27
00001100	12	00011100	28
00001101	13	00011101	29
00001110	14	00011110	30
00001111	15	00011111	31

1.7 十进制数转换为二进制数

为了把十进制转换为二进制数，将十进制数重复地除以2并保留余数。得到的第一个余数即为最低有效数字位（LSD），最后一个余数则为最高有效数字位（MSD）。



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

16

例1.3 把十进制数28_{10}转换为二进制数。

解 将该数重复地除以2并保留余数：

```
28/2 → 14 余数为0  (LSD)
14/2 → 7  余数为0
7/2  → 3  余数为1
3/2  → 1  余数为1
1/2  → 0  余数为1  (MSD)
```

17

则二进制数为11100_2。

例1.4 把十进制数65_{10}转换为二进制数。

解 将该数重复地除以2并保留余数：

```
65/2 → 32 余数为 1  (LSD)
32/2 → 16 余数为 0
16/2 → 8  余数为 0
8/2  → 4  余数为 0
4/2  → 2  余数为 0
2/2  → 1  余数为 0
1/2  → 0  余数为 1  (MSD)
```

则二进制数为1000001_2。

例1.5 把十进制数122_{10}转换为二进制数。

解 将该数重复地除以2并保留余数：

```
122/2 → 61 余数为0  (LSD)
61/2  → 30 余数为1
30/2  → 15 余数为0
15/2  → 7  余数为1
7/2   → 3  余数为1
3/2   → 1  余数为1
1/2   → 0  余数为1  (MSD)
```

则二进制数为1111010_2。

1.8 二进制数转换为十六进制数

为了把二进制数转换为十六进制数，将二进制数按每四位进行分组，然后计算出每组对应的十六进制数。如果二进制数不能准确地被分成四位一组的形式，则在二进制数的左边插入所需数量的0使得该二进制数刚好能分被为四位一组的形式。

18

例1.6 把二进制数10011111_2转换为十六进制数。

解 首先把二进制数分成四位一组的形式，然后计算出每一组对应的十六进制数：

```
10011111 = 1001 1111
              9    F
```

则十六进制数为$9F_{16}$。

例1.7 把二进制数1110111100001110_2转换为十六进制数。

解 首先把二进制数分成四位一组的形式，然后计算出每一组对应的十六进制数：

```
1110111100001110 = 1110 1111 0000 1110
                      E    F    0    E
```

则十六进制数为$EF0E_{16}$。



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

例1.8　把二进制数111110_2转换为十六进制数。

解　因为该二进制数不能被精确地分为四为一组的形式，在这种情况下，必须在这个数的左边插入两个0，这样这个二进制数就能被分为四位一组了：

```
111110 = 0011 1110
           3    E
```

则十六进制数为$3E_{16}$。

19

表1-2给出了十进制数对应的的十六进制数（0～31）。

表1-2　十进制数对应的十六进制数

十 进 制	十六进制	十 进 制	十六进制
0	0	16	10
1	1	17	11
2	2	18	12
3	3	19	13
4	4	20	14
5	5	21	15
6	6	22	16
7	7	23	17
8	8	24	18
9	9	25	19
10	A	26	1A
11	B	27	1B
12	C	28	1C
13	D	29	1D
14	E	30	1E
15	F	31	1F

1.9　十六进制数转换为二进制数

为了把十六进制数转换为二进制数，要计算出每个十六进制位对应的4位二进制数。

20

例1.9　把十六进制数$A9_{16}$转换位二进制数。

解　计算每位十六进制数对应的二进制数：

$A = 1010_2$　　$9 = 1001_2$

则二进制数为10101001_2。

例1.10　把十六进制数$FE3C_{16}$转换为二进制数。

解　计算每位十六进制数对应的二进制数：

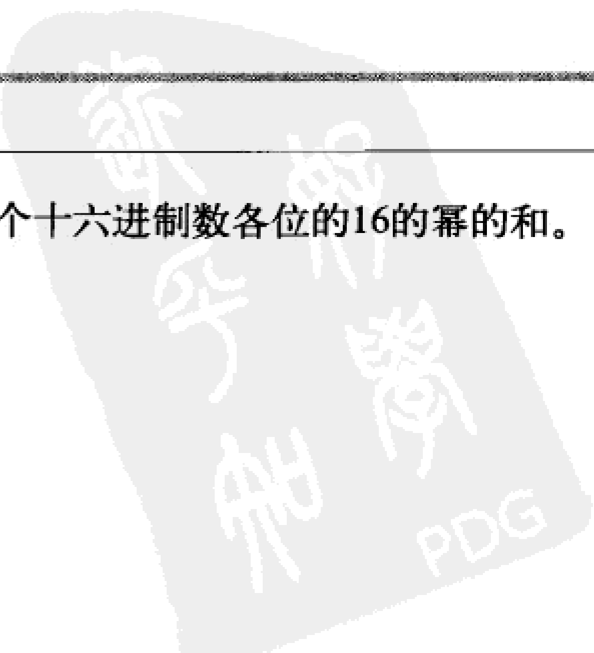
$F = 1111_2$　$E = 1110_2$　$3 = 0011_2$　$C = 1100_2$

则二进制数为1111111000111100_2。

1.10　十六进制数转换为十进制数

为了把十六进制数转换为二进制数，要计算出这个十六进制数各位的16的幂的和。

例1.11　把十六进制数$2AC_{16}$转换为十进制数。



世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

解 计算这个数各位的16的幂的和：

$$2AC_{16} = 2 \times 16^2 + 10 \times 16^1 + 12 \times 16^0$$
$$= 512 + 160 + 12$$
$$= 684$$

所得的十进制数为684_{10}。

21

例1.12 把十六进制数EE_{16}转换为十进制数。

解 计算这个数各位的16的幂的和：

$$EE_{16} = 14 \times 16^1 + 14 \times 16^0$$
$$= 224 + 14$$
$$= 238$$

所得的十进制数为238_{10}。

1.11 十进制数转换为十六进制数

为了把十进制数转换为十六进制数，将该十进制数重复地除以16并保留余数。第一个余数是LSD，最后一个余数是MSD。

例1.13 把十进制数238_{10}转换为十六进制数。

解 将该数重复地除以16得到余数：

```
238/16 → 14 余数为14（E） （LSD）
14/16  → 0  余数为14（E） （MSD）
```

则十六进制数为EE_{16}。

例1.14 把十进制数684_{10}转换为十六进制数。

解 将该数重复地除以16，得到余数：

```
684/16 → 42 余数为12（C） （LSD）
42/16  → 2  余数为10（A）
2/16   → 0  余数为 2       （MSD）
```

则十六进制数为$2AC_{16}$。

22

1.12 八进制数转换为十进制数

为了把一个八进制数转化为十进制数，要计算出这个八进制数各位的8的幂的和。

例1.15 把八进制数15_8转换为十进制数。

解 计算这个数各位的8的幂的和：

$$15_8 = 1 \times 8^1 + 5 \times 8^0$$
$$= 8 + 5$$
$$= 13$$

则十进制数为13_{10}。

例1.16 把八进制数237_8转换为十进制数。

解 计算这个数各位的8的幂的和：



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

$$
\begin{aligned}
237_8 &= 2 \times 8^2 + 3 \times 8^1 + 7 \times 8^0 \\
&= 128 + 24 + 7 \\
&= 159
\end{aligned}
$$

则十进制数为159_{10}。

1.13 十进制数转换为八进制数

为了把一个十进制数转换为八进制数，将该十进制数重复地除以8并保留余数。第一个余数是LSD，最后一个余数是MSD。

23

例1.17 把十进制数159_{10}转换为八进制数。

解 将该数重复地除以8，得到余数：

```
159/8   → 19  余数为7 （LSD）
19/8    → 2   余数为3
2/8     → 0   余数为2 （MSD）
```

则八进制数为237_8。

例1.18 把十进制数460_{10}转换为八进制数。

解 将该数重复地除以8，得到余数：

```
460/8   → 57  余数为4 （LSD）
57/8    → 7   余数为1
7/8     → 0   余数为7 （MSD）
```

则八进制数为714_8。

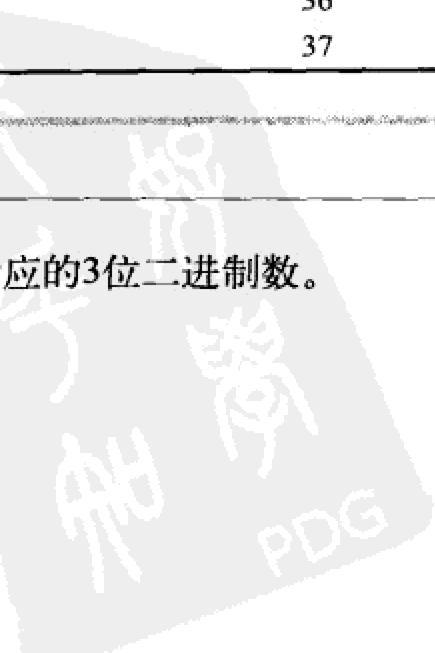
表1-3给出了十进制数对应的八进制数（0～31）。

表1-3 十进制数对应的八进制数

十进制	八进制	十进制	八进制
0	0	16	20
1	1	17	21
2	2	18	22
3	3	19	23
4	4	20	24
5	5	21	25
6	6	22	26
7	7	23	27
8	10	24	30
9	11	25	31
10	12	26	32
11	13	27	33
12	14	28	34
13	15	29	35
14	16	30	36
15	17	31	37

1.14 八进制数转换为二进制数

为了把八进制数转换为二进制数，要计算出每个八进制位对应的3位二进制数。



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

例1.19 把八进制数177_8转化为二进制数。

解 计算出每位八进制数对应的的二进制数：

```
1 = 001₂ 7 = 111₂ 7 = 111₂
```

则二进制数为001111111_2。

24

例1.20 把八进制数75_8转换为二进制数。

解 计算出每位八进制数对应的二进制数：

```
7 = 111₂ 5 = 101₂
```

则二进制数为111101_2。

25

1.15 二进制数转换为八进制数

为了把二进制数转换为八进制数，将二进制数分为三位一组的形式，然后计算出每一组对应的八进制数。

例1.21 把二进制数110111001_2转换为八进制数。

解 将该数分成三位一组的形式：

```
110111001 = 110 111 001
             6   7   1
```

则八进制数为671_8。

1.16 负数

二进制数的最高有效位通常用作符号位。按惯例，正数的符号位为0，负数的符号位为1。图1-5给出了4位的正数和负数。最大的正数和最小的负数分别是+7和−8。

26

二进制数	对应的十进制数
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

图1-5 4位的正数和负数

为了把一个正数转换为负数，求取原数的反码然后加1。这个过程也称作求2的补码。

例1.22 将十进制数−6用4位的二进制数表示。

解 首先把此数写成正数，然后算出反码再加1：

```
0110   +6
1001   反码
   1   加1
----
1010   结果是-6
```

例1.23 将十进制数−25用8位的二进制数表示。

解 首先把此数写成正数，然后算出反码再加1：

```
00011001   +25
11100110   补码
       1   加1
--------
11100111   结果是-25
```

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

1.17 二进制数的加法

二进制数的加法类似于十进制数的加法。将每一列的数及来自低位的进位相加。基本的加法运算如下：

27

```
0 + 0 = 1
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10          产生一个进位
1 + 1 + 1 = 11      产生一个进位
```

以下是一些例子。

例1.24 求二进制数011和110的和。

解 我们可以像十进制数的加法一样将这两个数相加。

```
  011     第1列：1 + 0 = 1
+ 110     第2列：1 + 1 = 0 产生一个进位
-----     第3列：1 + 1 = 10
1001
```

例1.25 求二进制数01000011和00100010的和。

解 我们可以像十进制数加法一样将这两个数相加。

```
  01000011    第1列：1 + 1 = 0
+ 00100010    第2列：1 + 1 = 10
----------    第3列：0 + 进位 = 1
  01100101    第4列：0 + 0 = 0
              第5列：0 + 0 = 0
              第6列：0 + 1 = 1
              第7列：1 + 0 = 1
              第8列：0 + 0 = 0
```

28

1.18 二进制数的减法

为了从一个二进制数中减去另一个二进制数，先把减数转化为负数，然后执行两数的加法。

例1.26 将二进制数0110减去0010。

解 首先把减数转化为负数：

```
0010   减数
1101   补码
   1   加1
----
1110
```

然后将两数相加：

```
  0110
+ 1110
--------
  0100
```

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

因为这里只使用了4位，所以看不出进位。

1.19 二进制数的乘法

两个二进制数的乘法类似于两个十进制数的乘法。这里有4种可能的形式：

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

以下是一些例子。

29

例1.27 求两个二进制数0110和0010的乘积。

解 将两数相乘如下：

```
   0110
   0010
   ----
   0000
  0110
 0000
0000
------
001100 或1100
```

本例中，最后的结果需要4位来表示。

例1.28 求二进制数1001和1010的乘积。

解 将两数相乘如下：

```
   1001
   1010
   ----
   0000
  1001
 0000
1001
------
1011010
```

本例中，最后的结果需要7位来表示。

30

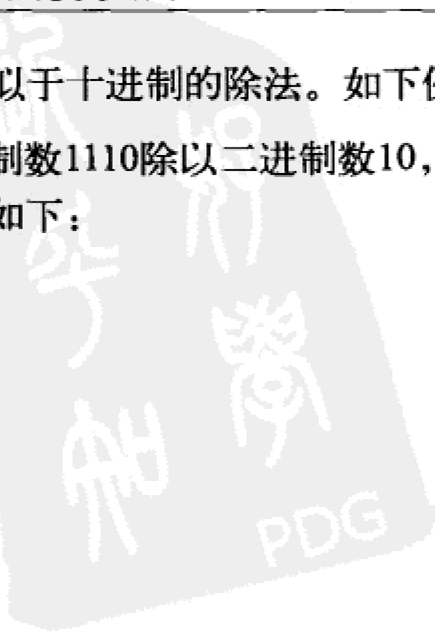
1.20 二进制数的除法

二进制的除法类似于十进制的除法。如下例所示。

例1.29 将二进制数1110除以二进制数10，求商。

解 将两数相除如下：

```
    111
```



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
10 ⌐1110
    10
    ----
     11
     10
    ----
      10
      10
    ----
      00
```

所得结果为111_2。

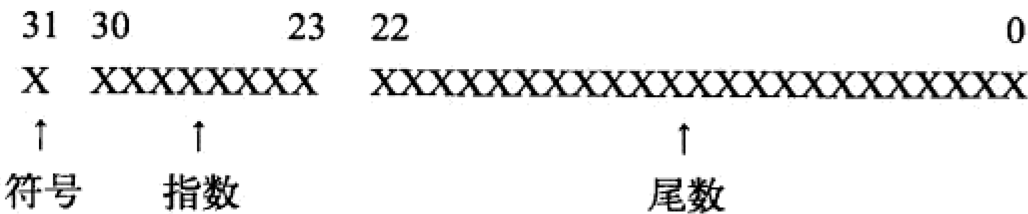
1.21 浮点数

浮点数是用来表示小数的，例如3.256、2.1、0.0036等。浮点数许多工程和技术计算中都很常用。最常见的浮点数标准是IEEE标准，浮点数可以用32位（单精度）或64位（双精度）来表示。

31

在本节中，我们只考虑32位格式的浮点数，并说明如何对这些数进行数学运算。

按照IEEE标准，32位的浮点数可表示如下：



最高位表示的是数的符号，0表示这是个正数，1表示这是个负数。

8位的指数表示这个数的幂。为了计算简单，没有给出指数的符号，而使用excess-128数制。因此，为了求得实际的指数，必须将给定的指数减去127。例如，若指数为10000000，则指数的实际值为128−127 = 1。

尾数是23位宽度的，表示2的负增长幂。例如，若尾数是11100000000000000000000，则尾数的值为$2^{-1}+2^{-2}+2^{-3}=7/8$。

一个浮点数的十进制对应数可用如下公式计算：

$$Number = (-1)^s 2^{e-127} 1.f$$

其中，$s=0$用于正数，$s=1$用于负数

$e=$指数（取值为0～255）

$f=$尾数

正如上述公式所示，在尾数前有一个隐藏的1（也就是，$1.f$表示尾数）。

用32位浮点形式表示的最大数为：

```
0 11111110 11111111111111111111111
```

该数是（$2-2^{-23}$）2^{127}，即十进制数3.403×10^{38}。这个数在十进制小数点后保留有多达六位的精度。

32

用32位浮点形式表示的最小数为：

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
0 00000001 00000000000000000000000
```

该数是2^{-126}，即十进制数1.175×10^{-38}。

1.22 浮点数转换为十进制数

为了把已知的浮点数转换为十进制数，必须找出浮点数的尾数和指数，然后转换为十进制数。

下面给出一些例子。

例1.30 计算浮点数0 10000001 10000000000000000000000对应的十进制数：

解

```
符号 = 正
指数 = 129-127 = 2
尾数 = 2^-1 = 0.5
```

该数对应的十进制数是$+1.5 \times 2^2 = +6.0$。

例1.31 计算浮点数0 10000010 11000000000000000000000对应的十进制数。

解 在这个例子中，有

```
符号 = 正
指数 = 130-127 = 3
尾数 = 2^-1 + 2^-2 = 0.75
```

因此，该数对应的十进制数是$+1.75 \times 2^3 = +14.0$。

33

1.22.1 规范化浮点数

浮点数通常以规范化形式表示。一个规范化的数在小数点前只有一位（假定在小数点前有隐藏的1）。

为了规范化一个给定的浮点数，必须重复地向左移动小数点，每移一位，则指数增加1。

下面给出一些例子。

例1.32 请将浮点数123.56规范化。

解 在小数点之前只保留一位数，重写该数可以得到：

1.2356×10^2

例1.33 请将二进制数1011.1_2规范化。

解 在小数点之前只保留一位数，重写该数可以得到：

1.0111×2^3

1.22.2 十进制数转换为浮点数

为了把已知的十进制数为转换为浮点数，可以执行如下的步骤:

- 写出该数的二进制形式；
- 将该数规范化；
- 计算尾数和指数；
- 写出该数的浮点数形式。

下面给出一些例子。

34

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

例1.34 请将十进制数2.25_{10}转换为浮点数。

解 写出该数的二进制形式：

$$2.25_{10} = 10.01_2$$

将该数规范化：

$$10.01_2 = 1.001_2 \times 2^1$$

这里，$s = 0$，$e - 127 = 1$，即$e = 128$，$f = 00100000000000000000000$。

（记住，假定在左边是存在数字1的，尽管这在计算中没有显示出来）。所求的浮点数可以写成：

```
s     e                  f
0  1000000     (1) 001 0000 0000 0000 0000
```

即所求的32位浮点数为：

```
01000000000100000000000000000000
```

例1.35 请将十进制数134.0625_{10}转换为浮点数。

解 写出该数的二进制形式：

$$134.0625_{10} = 10000110.0001_2$$

将该数规范化：

$$10000110.0001_2 = 1.00001100001 \times 2^7$$

35

这里，$s = 0$，$e - 127 = 7$，即$e = 134$，$f = 00001100001000000000000$。

所求的浮点数可以写成：

```
s      e                  f
0  10000110     (1) 00001100001000000000000
```

即所求的32位浮点数为：

```
01000011000001100001000000000000
```

1.22.3 浮点数的乘除法

浮点数的乘除法非常简单，具体操作步骤如下：

- 将这些数的指数相加（或相减）；
- 将这些数的尾数相乘（或相除）；
- 调整指数；
- 将该数规范化；
- 所得结果的符号是这两个数的符号位的EXOR运算。

因为指数在计算中被运算了两次，所以必须从指数中减去127。

下面是关于两个浮点数的乘法的例子。

例1.36 求十进制数0.5_{10}和0.75_{10}的浮点数，并计算它们的乘积。

解 将两个十进制数转化为浮点数：

$$0.5_{10} = 1.0000 \times 2^{-1}$$

这里，$s = 0$，$e - 127 = -1$，即$e = 126$，$f = 0000$，即

$$0.5_{10} = 0\ 01110110\ (1)\ 000\ 0000\ 0000\ 0000\ 0000\ 0000$$

类似地，

36

$$0.75_{10} = 1.1000 \times 2^{-1}$$

这里有$s = 0$，$e = 126$，$f = 1000$，即

PDG

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

$0.75_{10} = 0\ 01110110\ (1)100\ 0000\ 0000\ 0000\ 0000\ 0000$

将尾数相乘，得到结果为“(1) 100 0000 0000 0000 0000 0000”。指数的和为126 + 126 = 252。将指数减去127，可以得到252−127 = 125。对两数的符号位进行EXOR运算的结果为0。所以，浮点数形式的结果为：

```
0 01111101 (1)100 0000 0000 0000 0000 0000
```

该数对应于十进制数0.375（即0.5 × 0.75），这才是正确的结果。

1.22.4 浮点数的加减法

浮点数的指数必须相等才能进行加减运算。浮点数加减法的步骤可描述如下。

- 对较小的数进行右移位直到两数的指数相等。每次移位后，较小数的指数加1。
- 像整数运算一样将两数的尾数相加（或相减），无需考虑小数点。
- 将所得的结果规范化。

下面给出一个例子。

例1.37 请将十进制数0.5_{10}和0.75_{10}转换为浮点数，然后计算它们的和。

解 如例1.36所示，可以将这两个数转化为浮点数：

$0.5_{10} = 0\ 01110110\ (1)\ 000\ 0000\ 0000\ 0000\ 0000\ 0000$

类似地，

$0.75_{10} = 0\ 01110110\ (1)\ 100\ 0000\ 0000\ 0000\ 0000\ 0000$

因为两数的指数相同，所以无需对较小的数进行移位。若在不考虑小数点的情况下将它们的尾数相加，则可以得到：

```
(1) 000 0000 0000 0000 0000 0000
(1) 100 0000 0000 0000 0000 0000
                               +
--------------------------------
(10)100 0000 0000 0000 0000 0000
```

对得到的结果规范化，将该数右移一位，然后将它的指数加1，则最终的结果为：

```
0 01111111 (1)010 0000 0000 0000 0000 0000
```

该浮点数等于十进制数1.25，即十进制数0.5和0.75的和。

将浮点数转换为十进制数和将十进制数转换为浮点数的程序可以在下面的网址免费得到：

http://babbage.cs.qc.edu/courses/cs341/IEEE-754.html

37

1.23 BCD 数

BCD（binary coded decimal，二进制编码的十进制）数常用于显示系统，如LCD和7段数码显示器。在BCD数中，每个数字（从0到9）都是一个4位的数。作为例子，表1-4列出了0～20的BCD数。

表1-4 0～20的BCD数

十进制数	BCD	二进制数
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

（续）

十进制数	BCD	二进制数
7	0111	0111
8	1000	1000
9	1001	1001
10	0001 0000	1010
11	0001 0001	1011
12	0001 0010	1100
13	0001 0011	1101
14	0001 0100	1110
15	0001 0101	1111
16	0001 0110	1 0000
17	0001 0111	1 0001
18	0001 1000	1 0010
19	0001 1001	1 0011
20	0010 0000	1 0100

例1.38 将十进制数295写成BCD数。

38~39

解 写出每个数字对应的4位二进制数：

$2 = 0010_2$, $9 = 1001_2$, $5 = 0101_2$

则BCD数为$0010\ 1001\ 0101_2$。

例1.39 请写出BCD数$1001\ 1001\ 0110\ 0001_2$对应的十进制数。

解 写成每4位一组BCD数的对应十进制数，于是得到的十进制数为9961。

1.24 小结

第1章介绍了微处理器和微控制器系统，简要地描述了微控制器的基本构造。本章还介绍了几种数制，描述了不同数制之间的相互转换。以举例的形式阐述了浮点数和浮点数运算的内容。

1.25 练习题

1. 什么是微控制器？什么是微处理器？说明微控制器和微处理器之间的主要区别。
2. 找出你身边的一些微控制器应用。
3. 在哪里会用到EPROM存储器？
4. 在哪里会用到RAM存储器？
5. 说明微控制器通常使用的存储器类型。
6. 什么是输入/输出端口？
7. 什么是模数转换器？举例说明模数转换器是如何工作的。
8. 说明看门狗定时器在实时系统中的用途。
9. 什么是串行输入/输出？在哪里会用到串行通信？

40

10. 在输出端口引脚的性能指标中，为什么电流拉出/灌入能力非常重要呢？
11. 什么是中断？微控制器识别到中断时将会发生什么？
12. 为什么掉电检测在实时系统中非常重要？
13. 说明基于RISC的微控制器和基于CISC的微控制器的区别。PIC是哪种类型的微控制器？
14. 将下面的十进制数转换为二进制数。

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

a) 23　　b) 128　　c) 255　　d) 1023
e) 120　　f) 32000　　g) 160　　h) 250

15. 将下面的二进制数转换为十进制数。
a) 1111　　b) 0110　　c) 11110000
d) 00001111　　e) 10101010　　f) 10000000

16. 将下面的八进制数转换为十进制数。
a) 177　　b) 762　　c) 777　　d) 123
e) 1777　　f) 655　　g) 177777　　h) 207

17. 将下面的十进制数转换为八进制数。
a) 255　　b) 1024　　c) 129　　d) 2450
e) 4096　　f) 256　　g) 180　　h) 4096

18. 将下面的十六进制数转换为十进制数。
a) AA　　b) EF　　c) 1FF　　d) FFFF
e) 1AA　　f) FEF　　g) F0　　h) CC

19. 将下面的二进制数转换为十六进制数。
a) 0101　　b) 11111111　　c) 1111　　d) 1010
e) 1110　　f) 10011111　　g) 1001　　h) 1100

20. 将下面的二进制数转换为八进制数。
a) 111000　　b) 000111　　c) 1111111　　d) 010111
e) 110001　　f) 11111111　　g) 1000001　　h) 110000

41

21. 将下面的八进制数转换为二进制数。
a) 177　　b) 7777　　c) 555　　d) 111
e) 1777777　　f) 55571　　g) 171　　h) 1777

22. 将下面的十六进制数转换为八进制数。
a) AA　　b) FF　　c) FFFF　　d) 1AC
e) CC　　f) EE　　g) EEFF　　h) AB

23. 将下面的八进制数转换为十六进制数。
a) 177　　b) 777　　c) 123　　d) 23
e) 111　　f) 17777777　　g) 349　　h) 17

24. 将下面的十进制数转换为浮点数。
a) 23.45　　b) 1.25　　c) 45.86　　d) 0.56

25. 将下面的十进制数转换为浮点数，然后计算它们的和。
0.255和1.75

26. 将下面的十进制数转换为浮点数，然后计算它们的积。
2.125和3.75

27. 将下面的十进制数转换为BCD数。
a) 128　　b) 970　　c) 900　　d) 125

42

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

第 2 章　PIC18F 系列微控制器

PIC16系列微控制器已经流行很多年了。虽然这些都是非常出色的通用微控制器，但是也有一定的局限性。例如，程序存储器和数据存储器的容量很有限，栈也很小，中断结构很简单，所有的中断源共享同一个中断向量。PIC16系列微控制器也不直接支持诸如USB、CAN总线等高级外围接口，并且与这些设备的接口连接也不是很容易。这些微控制器的指令集也很有限。例如，它们没有乘法指令和除法指令，而且分支指令也很简单，仅仅是skip指令和goto指令的组合。

Microchip公司已成功开发出PIC18系列微控制器，用于引脚数多、密度高的复杂应用。PIC18F微控制器提供性价比高的解决方案，用于使用实时操作系统（RTOS）和需要诸如TCP/IP、CAN、USB或者ZigBee这样复杂的通信协议栈，且用C语言实现通用应用系统。PIC18F设备提供从8 KB到128 KB不等的闪存和从256 B到4 KB不等的数据存储器，工作电压范围是2.0 V～5.0 V，频率是直流到40 MHz。

PIC18F系列微控制器的基本特性如下：

- 77条指令
- 可兼容PIC16微控制器的源代码
- 程序存储器寻址范围多达2 MB
- 数据存储器寻址范围多达4 KB
- 工作频率在从直流到40 MHz之间
- 8×8硬件乘法器
- 带有中断优先级
- 16位宽指令、8位宽数据通道
- 多达2个8位的定时器/计数器
- 多达3个16位的定时器/计数器
- 多达4个外部中断
- 强大的电流拉出（25 mA）/灌入能力
- 多达5个捕捉/比较/PWM模块
- 主同步串行端口模块（SPI和I^2C模式）
- 多达2个USART模块
- 并行从端口（PSP）
- 快速的10位模数转换器
- 可编程的低压检测（LVD）模块
- 上电复位（POR）、上电定时器（PWRT）和振荡器起振定时器（OST）
- 带有片上RC振荡器的看门狗定时器（WDT）
- 内电路编程

另外，一些PIC18F系列的微控制器具有下面的特殊性能：

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

- 直接的CAN2.0总线接口
- 直接的USB2.0总线接口
- 直接的LCD控制接口
- TCP/IP接口
- ZigBee接口
- 直接的电机控制接口

44

在PIC18F系列中，大多数设备都是相互兼容的。表2-1给出了该系列中一些比较流行的设备的特点。本章将详细描述和研究PIC18FXX2微控制器。PIC18F系列中大多数其他微控制器的结构都是与之相似的。

表2-1　PIC18FXX2微控制器系列

特　性	PIC18F242	PIC18F252	PIC18F442	PIC18F452
程序存储器（B）	16K	32K	16K	32K
数据存储器（B）	768	1 536	768	1 536
EEPROM（B）	256	256	256	256
I/O端口	A，B，C	A，B，C	A，B，C，D，E	A，B，C，D，E
定时器	4	4	4	4
中断源	17	17	18	18
捕捉/比较/脉宽调制模块（PWM）	2	2	2	2
串行通信	MSSP USART	MSSP USART	MSSF USART	MSSP USART
A/D转换器（10位）	5通道	5通道	8通道	8通道
低压检测	是	是	是	是
掉电复位	是	是	是	是
封装	28-引脚DIP 28-引脚SOIC	28-引脚DIP 28-引脚SOIC	40-引脚DIP 44-引脚PLCC 44-引脚TQFP	40-引脚DIP 44-引脚PLCC 44-引脚TQFP

读者可能对PIC16F系列的编程和应用会比较熟悉。在开始深入学习PIC18F系列之前，很有必要比较一下PIC18F系列和PIC16F系列各自的特性。

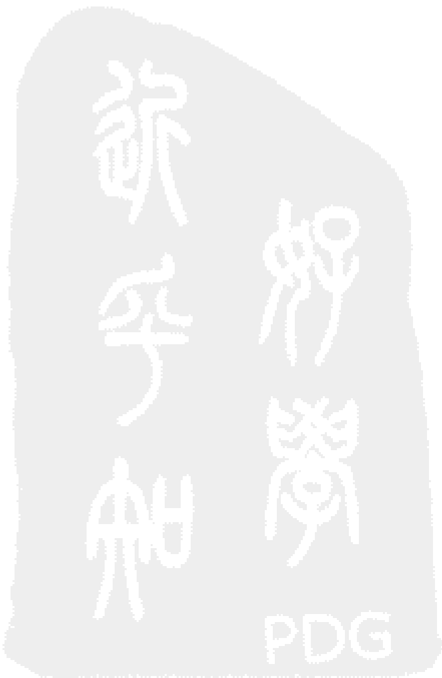
下面是PIC16F系列和PIC18F系列之间相似的特性：

- 相似的封装和引脚
- 相似的特殊功能寄存器（SFR）名字和功能
- 相似的外围设备
- PIC18F指令系统的子集
- 相似的开发工具

45

下面是PIC18F系列的新特性：

- 指令的数量增加1倍
- 16位的指令字
- 硬件8×8乘法器
- 更多的外部中断
- 基于优先级的中断
- 更强的状态寄存器
- 更大的程序存储器和数据存储器空间



世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

- 更大的栈
- 锁相环（PLL）时钟发生器
- 更强的输入输出结构
- 配置寄存器集
- 更快的运行速度
- 更低的功耗

2.1　PIC18FXX2的结构

由表2-1可以看出，PIC18FXX2系列包括4个器件。PIC18F2X2微控制器是28引脚的器件，而PIC18F4X2微控制器是40引脚的器件。除了较大的器件具有更多的输入输出端口和更多的A/D转换器通道之外，两类微控制器的结构几乎是相同的。在这部分，我们将详细介绍PIC18F452微控制器的结构。其他的标准PIC18F系列微控制器的结构很相似，在这部分学习到的知识应该足以帮助我们理解其他PIC18F系列微控制器的工作原理。

PIC18F452微控制器（DIP封装）的引脚配置如图2-1所示。这是一个具有DIL形式封装的40引脚微控制器，其引脚配置和流行的PIC16F877微控制器很相似。

46

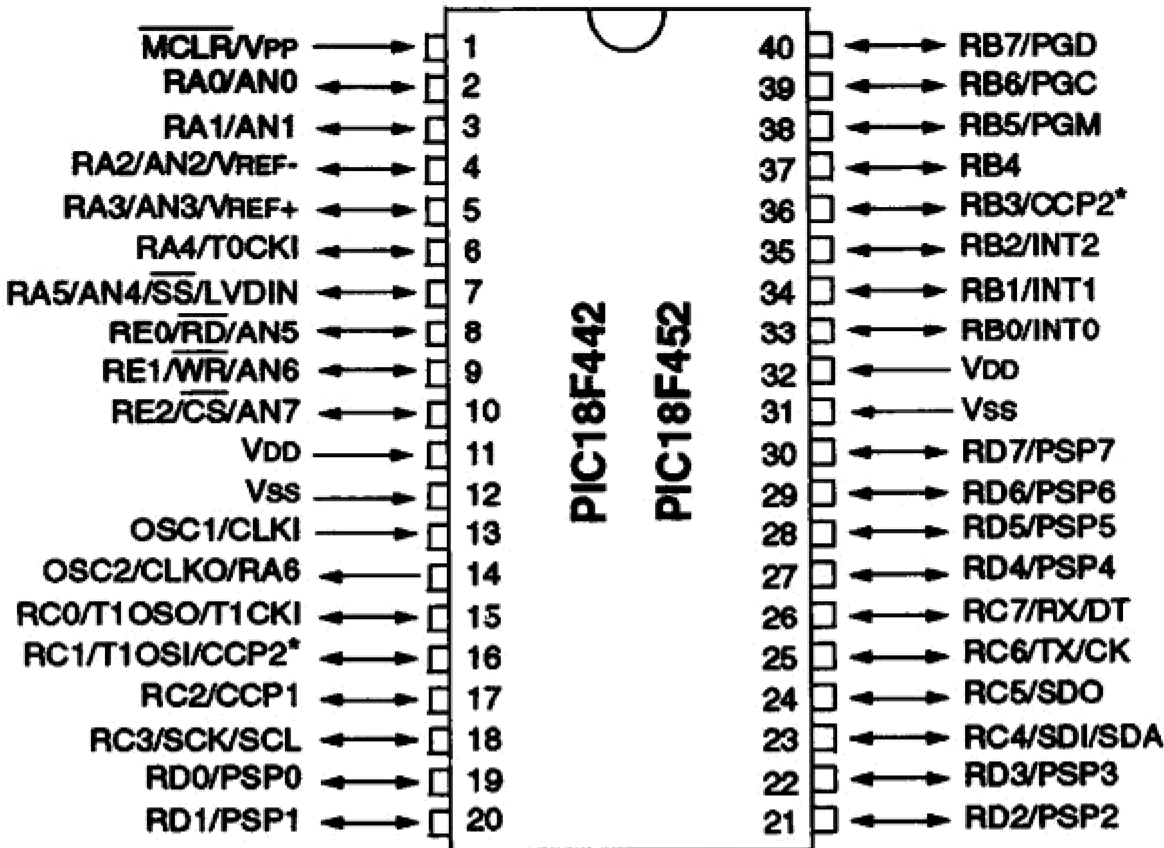
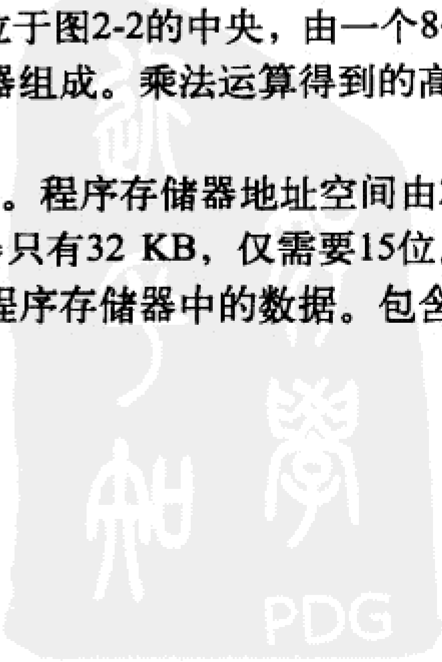


图2-1　PIC18F452微控制器的DIP引脚配置

图2-2是PIC18F452微控制器的内部结构图。CPU位于图2-2的中央，由一个8位ALU、一个8位的工作累加寄存器（WREG）和一个8×8硬件乘法器组成。乘法运算得到的高字节和低字节被分别存储在PRODH和PRODL两个8位的寄存器中。

在图2-2的左上方部分是程序计数器和程序存储器。程序存储器地址空间由21位的2 MB程序存储器单元组成。PIC18F452微控制器的程序存储器只有32 KB，仅需要15位。剩下的6位地址是多余的，且没有被使用。使用表指针能访问表和程序存储器中的数据。包含31级栈的程序存储器通常用来存储中断和子程序返回地址。



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

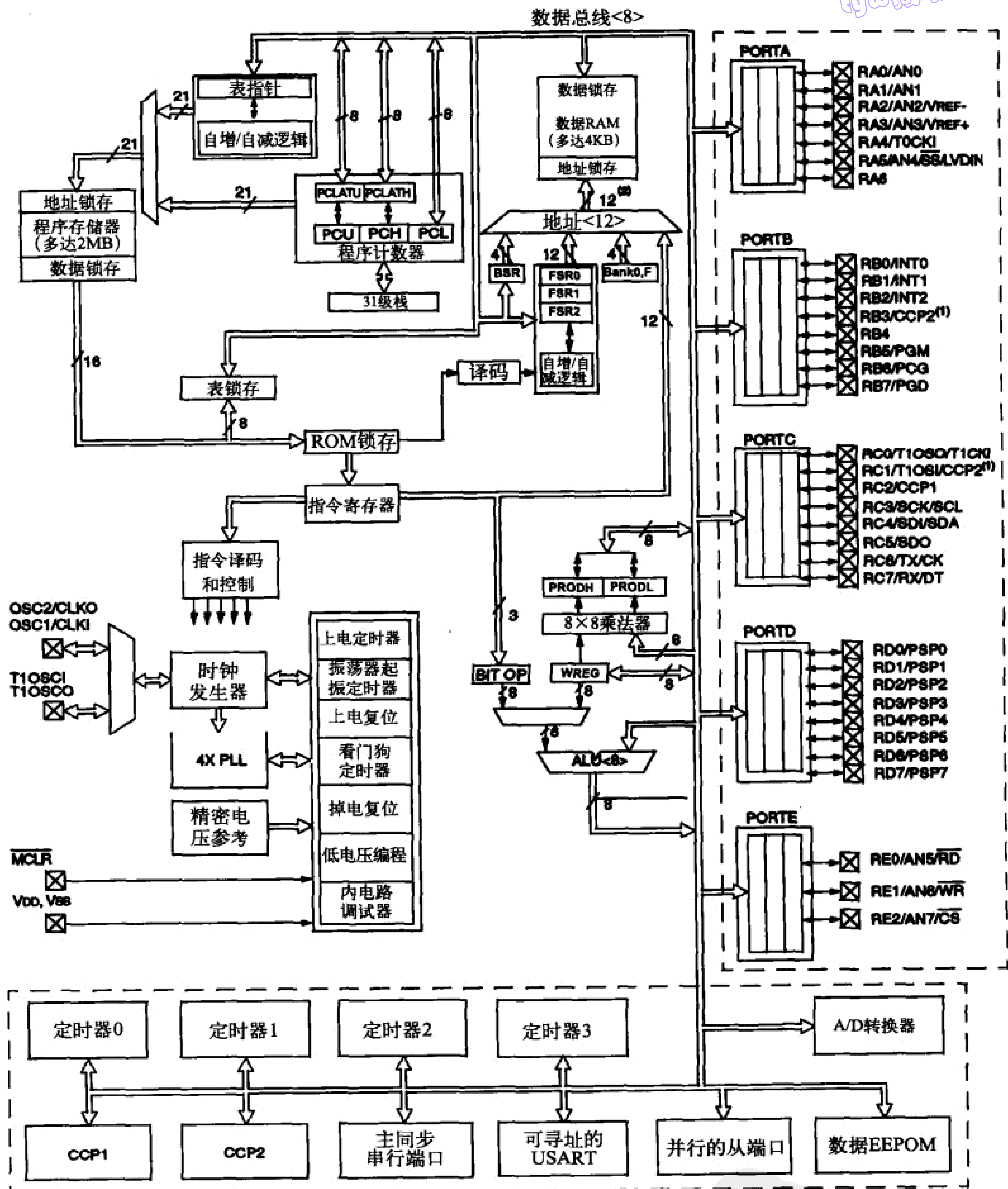


图2-2 PIC18F452微控制器的内部结构图

在图2-2的正上方是数据存储器。数据存储器的总线宽度是12位，可以访问4 KB的数据存储器单元。接下来将看到，数据存储器由特殊功能寄存器（SFR）和通用寄存器组成，并以存储区的形式进行排列。

图2-2的底部是定时器/计数器、捕捉/比较/PWM寄存器、USART、A/D 转换器和EEPOM数据存储器。PIC18F452包括：

47~48

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

- 4个定时器/计数器
- 2个捕捉/比较/PWM模块
- 2个串行通信模块
- 8个10位A/D 转换器通道
- 256 B EEPROM

图2-2的左边是振荡器电路，包括：

- 上电定时器
- 振荡器起振定时器
- 上电复位
- 看门狗定时器
- 掉电复位
- 低电压编程
- 内电路调试
- PLL电路
- 时钟发生电路

PLL电路是PIC18F系列的全新特性，提供倍乘振荡频率的选择以提高微控制器的运算速度。当遇到程序故障时，看门狗定时器能够被用来强制微控制器重新启动。当微控制器正在执行一个程序时，内电路调试器在程序开发期间非常有用，可用来返回包括寄存器值在内的诊断数据。

在图2-2的右边是输入输出端口。PIC18F452微控制器有五个并行端口，名字分别是PORTA、PORTB、PORTC、PORTD和PORTE。大多数端口引脚都是多功能的。例如，PORTA端口的引脚可用作并行输入输出或者模拟输入。PORTB端口的引脚可用作并行输入输出或者中断输入。

49

2.1.1 程序存储器结构

程序存储区分配图如图2-3所示。所有的PIC18F器件都有一个21位宽的程序计数器，因此可寻址最大为2MB的程序存储空间。PIC18F452微控制器上的用户存储区空间是从00000H到7FFFH。访问不存在的存储器空间（从8000H到1FFFFFH），所读得的值全部为0。当上电复位后程序开始执行时，复位向量地址为0000。地址0008H和0018H是分别预留给高优先级和低优先级中断向量的，并且写入的中断服务程序必须从其中的一个存储单元开始执行。

50

PIC18F微控制器具有一个31级的栈，用来存储子程序调用或者中断处理的返回地址。该栈

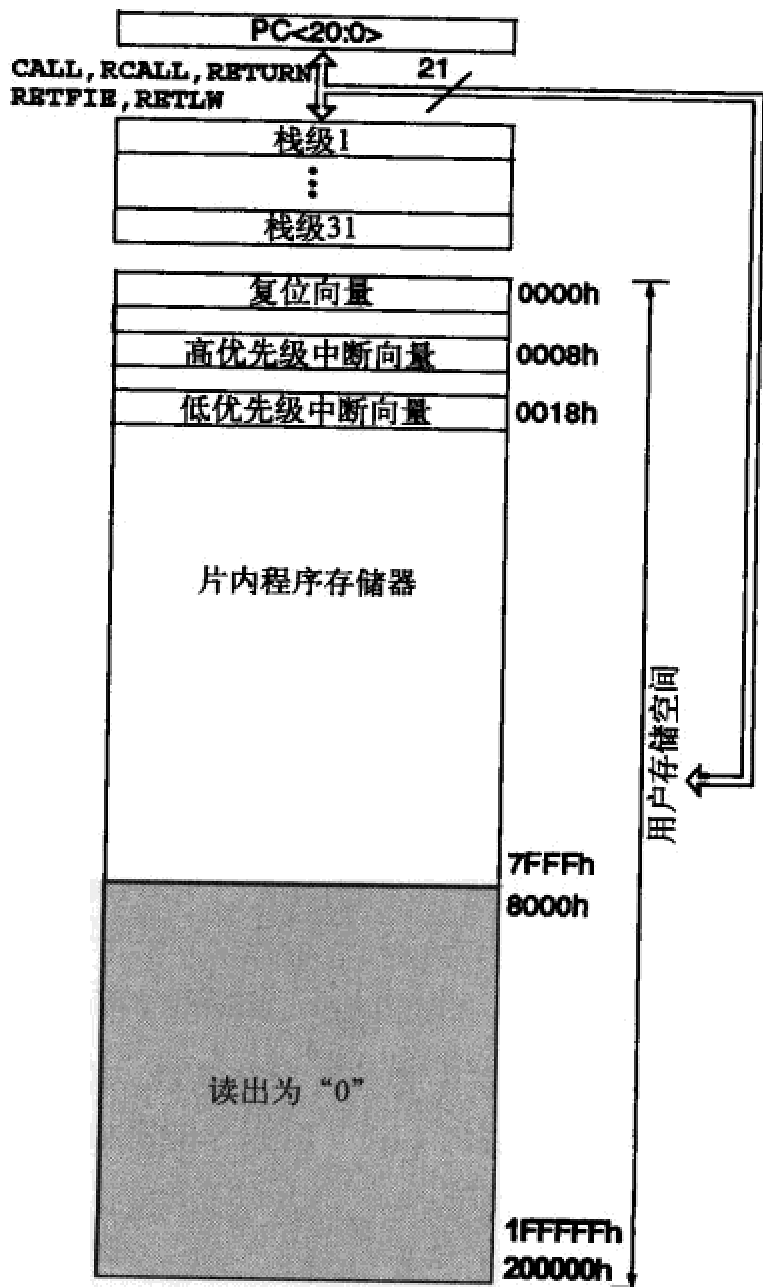


图2-3 PIC18F452微控制器的程序存储区分配图

PDG

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

并不是程序或数据存储器空间的一部分，它由一个5位的栈指针控制，其在微控制器复位后的初始值为00000。在子程序调用（或者中断）中，栈指针首先自动增量，它所指向的内存单元是写入程序计数器的内容。在子程序调用（或者中断）的返回过程中，该栈指针所指向的内存单元地址是自动减量的。在本书中，所有的项目都是基于C语言的。由于子程序和中断调用/返回操作都是由C语言的编译器自动完成的，所以这些操作在这里就不再详细介绍了。

程序存储器以字节（B）为单位，而指令在程序存储器中的单位则是2 B或者4 B。一条指令语句的最低有效字节通常都存储在程序存储器的偶地址。

一条指令周期包括4个周期：读取周期从程序计数器在Q1周期的递增开始；在执行周期中，所取得的指令在Q1周期被锁存到指令寄存器中；然后，该指令在Q2、Q3和Q4周期内被译码并执行；数据存储器单元在Q2周期内被读取，并在Q4周期内被写入。

2.1.2 数据存储器结构

PIC18F452微控制器的数据存储器存储区分配图如图2-4所示。数据存储器地址总线为12位，地址容量达到4 MB。通常，存储器由16个存储区组成，每个存储区为256 B，其中只使用了6个存储区。PIC18F452微控制器具有1 536 B的数据存储空间（6个存储区 × 256 B/每个存储区），位于数据存储器器的低端。当使用高级语言编译器时，存储区会自动地转换，因此用户在编程的时候不必担心存储区的选择问题。

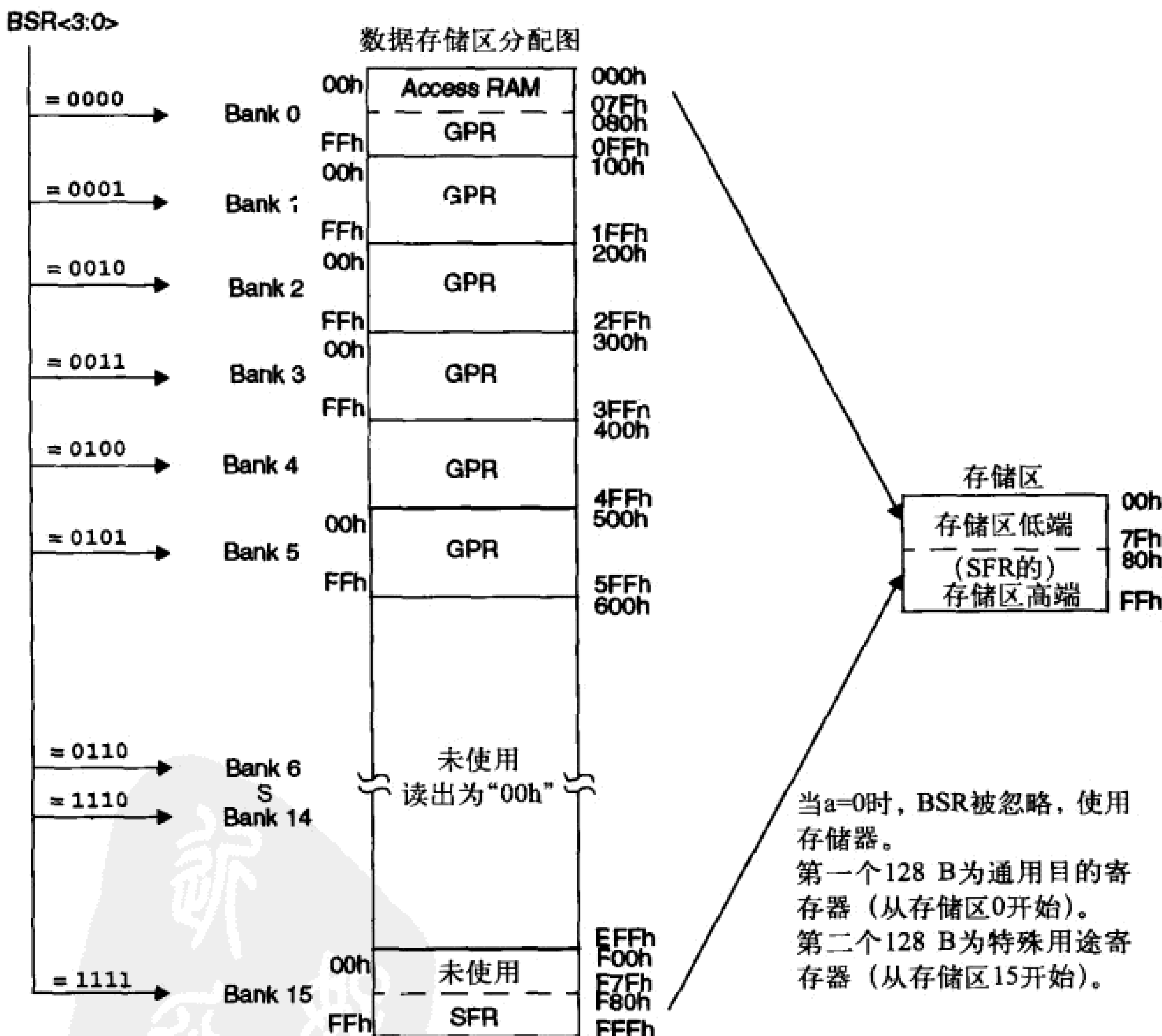


图2-4 PIC18F452微控制器的数据存储区分配图

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

特殊功能寄存器（SFR）占用数据存储区的顶端。SFR含有控制诸如外围设备、定时器/计数器、A/D转换器、中断和USART等操作的寄存器。PIC18F452微控制器的SFR寄存器如图2-5所示。

51

地址	名称	地址	名称	地址	名称	地址	名称
FFFh	TOSU	FDFh	INDF2[3]	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2[3]	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2[3]	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2[3]	FBCh	CCPR2H	F9Ch	—
FFBh	PCLATU	FDBh	PLUSW2[3]	FBBh	CCPR2L	F9Bh	—
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—
FF9h	PCL	FD9h	FSR2L	FB9h	—	F99h	—
FF8h	TBLPTRU	FD8h	STATUS	FB8h	—	F98h	—
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	—	F97h	—
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—	F96h	TRISE[2]
FF5h	TABLAT	FD5h	T0CON	FB5h	—	F95h	TRISD[2]
FF4h	PRODH	FD4h	—	FB4h	—	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—
FF0h	INTCON3	FD0h	RCON	FB0h	—	F90h	—
FEFh	INDF0[3]	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—
FEEh	POSTINC0[3]	FCEh	TMR1L	FAEh	RCREG	F8Eh	—
FEDh	POSTDEC0[3]	FCDh	T1CON	FADh	TXREG	F8Dh	LATE[2]
FECh	PREINC0[3]	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD[2]
FEBh	PLUSW0[3]	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—
FE7h	INDF1[3]	FC7h	SSPSTAT	FA7h	EECON2	F87h	—
FE6h	POSTINC1[3]	FC6h	SSPCON1	FA6h	EECON1	F86h	—
FE5h	POSTDEC1[3]	FC5h	SSPCON2	FA5h	—	F85h	—
FE4h	PREINC1[3]	FC4h	ADRESH	FA4h	—	F84h	PORTE[2]
FE3h	PLUSW1[3]	FC3h	ADRESL	FA3h	—	F83h	PORTD[2]
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	—	FA0h	PIE2	F80h	PORTA

图2-5　PIC18F452微控制器的SFR寄存器

2.1.3　配置寄存器

PIC18F452微控制器有一组配置寄存器（而PIC16系列微控制器只有一个配置寄存器）。在使用编程设备对闪存进行编程期间，执行对配置寄存器的编程。表2-2列出了这些寄存器，而表2-3给出了其具体描述。本节将详细介绍几个重要的配置寄存器。

52

1. CONFIG1H

CONFIG1H配置寄存器的地址是300001H，被用来选择微控制器的时钟源。CONFIG1H配置寄存器的位模式如图2-6所示。

53

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

表2-2 PIC18F452微控制器的配置寄存器

文件名		位7	位6	位5	位4	位3	位2	位1	位0	默认/未给定值
300001h	CONFIG1H	—	—	$\overline{\text{OSCSEN}}$	—	—	FOSC2	FOSC1	FOSC0	--1--111
300002h	CONFIG2L	—	—	—	—	BORV1	BORV0	BOREN	$\overline{\text{PWRTEN}}$	----1111
300003h	CONFIG2H	—	—	—	—	WDTPS2	WDTPS1	WDTPS0	WDTEN	----1111
300005h	CONFIG3H	—	—	—	—	—	—	—	CCP2MX	-------1
300006h	CONFIG4L	$\overline{\text{DEBUG}}$	—	—	—	—	LVP	—	STVREN	1--- -1-1
300008h	CONFIG5L	—	—	—	—	CP3	CP2	CP1	CP0	----1111
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	11-- ----
30000Ah	CONFIG6L	—	—	—	—	WRT3	WRT2	WRT1	WRT0	----1111
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—	111- ----
30000Ch	CONFIG7L	—	—	—	—	EBTR3	EBTR2	EBTR1	EBTR0	----1111
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	-1-- ----
3FFFFEh	DEVID1	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	(1)
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0000 0100

说明：x=未知，u=未变，－=未实现，q=值，要依据条件而定，阴影单元是未实现的，读出为“0”。

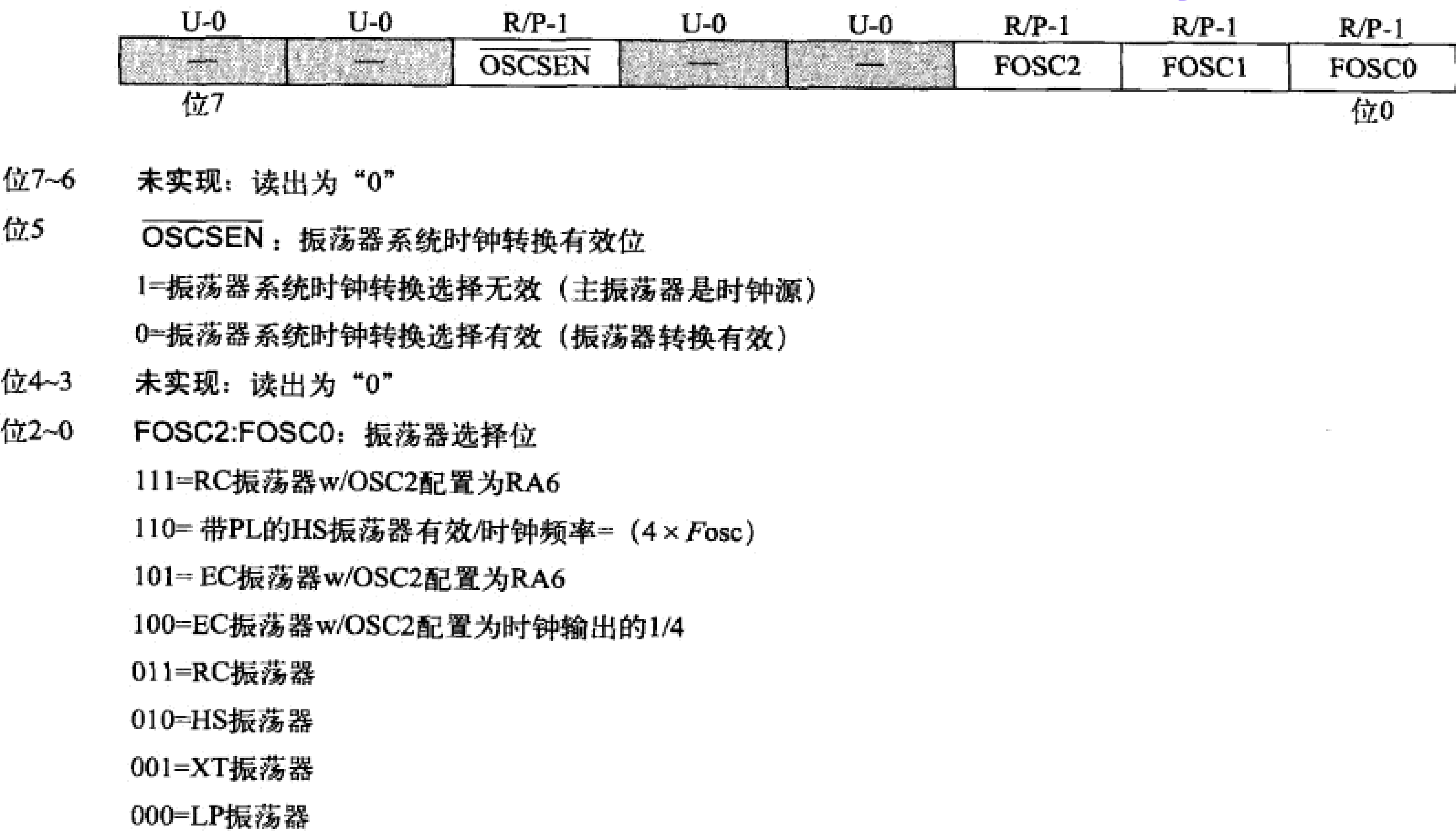
表2-3 PIC18F452微控制器的配置寄存器描述

配 置 位	描 述
OSCSEN	时钟脉冲源转换有效
FOSC2：FOSC0	振荡器模式
BORV1：BORV0	掉电复位电压
BOREN	掉电复位有效
PWRTEN	上电定时器有效
WDTPS2：WDTPS0	看门狗定时器后分频位
WDTEN	看门狗定时器有效
CCP2MX	CCP2多路复用器
DEBUG	调试有效
LVP	低电压编程有效
STVREN	栈满/下溢复位有效
CP3：CP0	代码保护
CPD	EEPROM代码保护
CPB	引导块代码保护
WRT3：WRT0	程序存储器写入保护
WRTD	EEPROM写入保护
WRTB	引导块写入保护
WRTC	配置寄存器写入保护
EBTR3：EBTR0	表读取保护
EBTRB	引导块表读取保护
DEV2：DEV0	设备ID位（001=18F452）
REV4：REV0	校正ID位
DEV10：DEV3	设备ID位

54 ~ 55

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源

tyw藏书



U-0	U-0	R/P-1	U-0	U-0	R/P-1	R/P-1	R/P-1
—	—	$\overline{\text{OSCSEN}}$	—	—	FOSC2	FOSC1	FOSC0
位7							位0

位7~6　未实现：读出为“0”

位5　$\overline{\text{OSCSEN}}$：振荡器系统时钟转换有效位
1=振荡器系统时钟转换选择无效（主振荡器是时钟源）
0=振荡器系统时钟转换选择有效（振荡器转换有效）

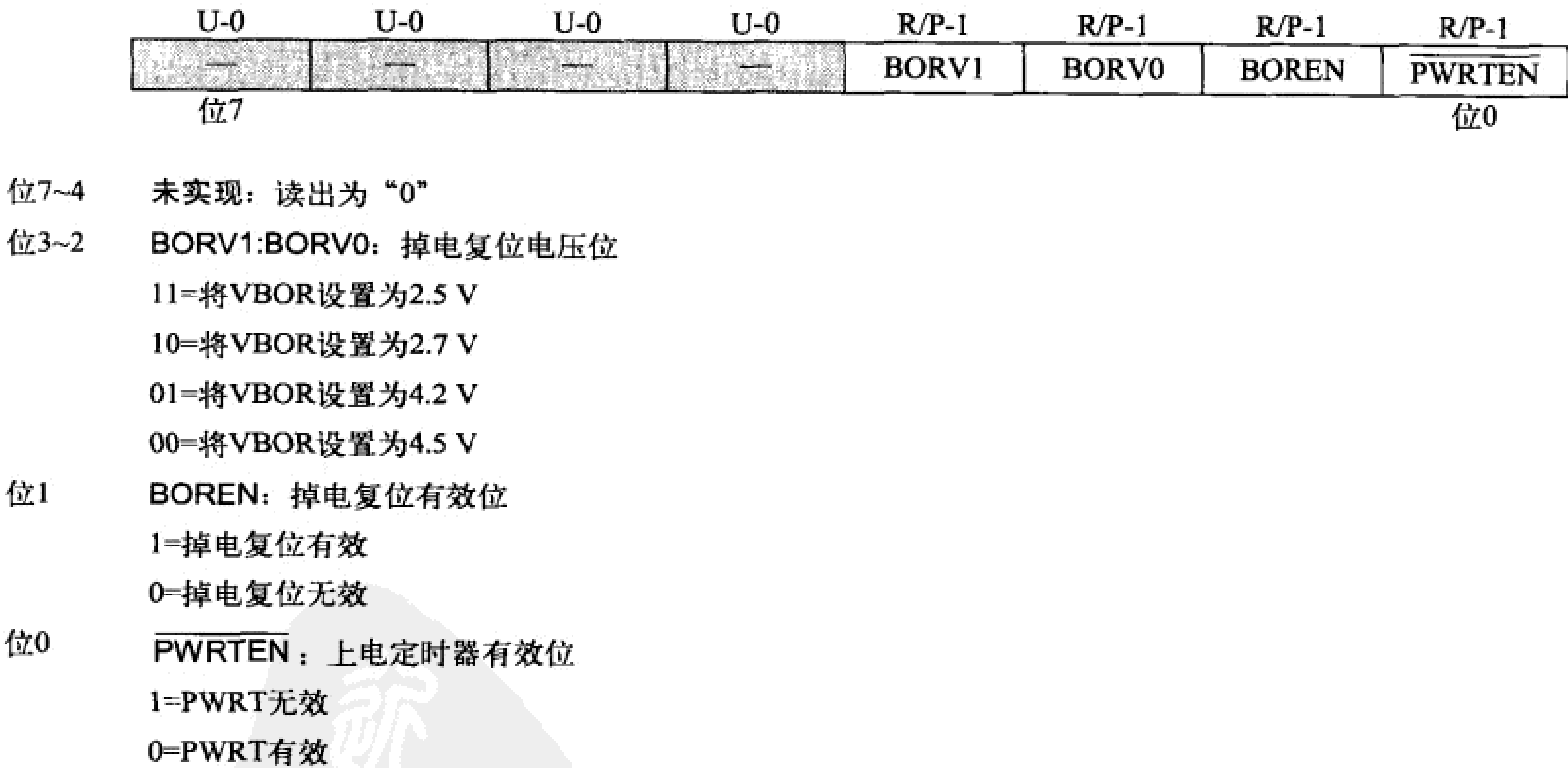
位4~3　未实现：读出为“0”

位2~0　FOSC2:FOSC0：振荡器选择位
111=RC振荡器w/OSC2配置为RA6
110= 带PL的HS振荡器有效/时钟频率=（4×*Fosc*）
101= EC振荡器w/OSC2配置为RA6
100=EC振荡器w/OSC2配置为时钟输出的1/4
011=RC振荡器
010=HS振荡器
001=XT振荡器
000=LP振荡器

图2-6　CONFIG1H寄存器的位模式

2. CONFIG2L

CONFIG2L配置寄存器的地址为300002H，被用来选择掉电电压位。CONFIG2L配置寄存器的位模式如图2-7所示。



U-0	U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	—	BORV1	BORV0	BOREN	$\overline{\text{PWRTEN}}$
位7							位0

位7~4　未实现：读出为“0”

位3~2　BORV1:BORV0：掉电复位电压位
11=将VBOR设置为2.5 V
10=将VBOR设置为2.7 V
01=将VBOR设置为4.2 V
00=将VBOR设置为4.5 V

位1　BOREN：掉电复位有效位
1=掉电复位有效
0=掉电复位无效

位0　$\overline{\text{PWRTEN}}$：上电定时器有效位
1=PWRT无效
0=PWRT有效

56

图2-7　CONFIG2L寄存器的位模式

3. CONFIG2H

CONFIG2H配置寄存器的地址为300003H，被用来选择看门狗操作。CONFIG2H配置寄存器的位模式如图2-8所示。

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

U-0	U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	—	WDTPS2	WDTPS1	WDTPS0	WDTEN
位7							位0

位7~4　**未实现：读出为"0"**

位3~1　**WDTPS2:WDTPS0：看门狗定时器后分频选择位**

111=1:128

110=1:64

101=1:32

100=1:16

011=1:8

010=1:4

001=1:2

000=1:1

位0　**WDTEN：看门狗定时器有效位**

1= WDT有效

0= WDT无效（由SWDTEN位进行控制）

图2-8　CONFIG2H寄存器位

PIC18LFXX2（工业的）			**标准的工作操作条件（除非有特殊说明）** 工作温度　对于工业应用为　−40℃ ≤ T_A ≤ +85℃				
PIC18FXX2（工业的，扩展的）			**标准的工作操作条件（除非有特殊说明）** 工作温度　对于工业应用为 −40℃ ≤ T_A ≤ +85℃ 对于扩展应用为 −40℃ ≤ T_A ≤ +125℃				
参数编号	**符号**	**特点**	**最小值**	**类型**	**最大值**	**单位**	**条件**
	V_{DD}	**电源电压**					
D001		PIC18LFXX2	2.0	—	5.5	V	HS、XT、RC和LP振荡器模式
D001		PIC18FXX2	4.2	—	5.5	V	
D002	V_{DR}	RAM数据保存电压	1.5	—	—	V	
D003	V_{POR}	V_{DD}起动电压 保证内部上电复位信号	—	—	0.7	V	详情请查阅3.1节（上电复位）
D004	S_{VDD}	V_{DD}上升速度 保证内部上电复位信号	0.05	—	—	V/ms	详情请查阅3.1节（上电复位）
D005	V_{BOR}	掉电复位电压					
		PIC18LFXX2					
		BORV1：BORV0=11	1.98	—	2.14	V	85℃ ≥ T ≥ 25℃
		BORV1：BORV0=10	2.67	—	2.89	V	
		BORV1：BORV0=01	4.16	—	4.5	V	
		BORV1：BORV0=00	4.45	—	4.83	V	
D005		PIC18FXX2					
		BORV1：BORV0=1x	不适用	—	不适用	V	不在设备的工作电压范围内
		BORV1：BORV0=01	4.16	—	4.5	V	
		BORV1：BORV0=00	4.45	—	4.83	V	

说明：给行加上阴影是为了方便读者阅读本表。

图2-9　PIC18F452微控制器的工作电源参数

PDG

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

2.1.4　电源

PIC18F452微控制器的工作电源需求如图2-9所示。正如图2-10所示，PIC18F452微控制器能够在电源电压为4.2 V～5.5 V的情况下以40 MHz的频率全速运行。作为更低功率的微控制器，PIC18LF452能够在2.0 V～5.5 V下工作。在更低的电压下，微控制器的最大时钟频率为4 MHz，而在工作电压为4.2 V时可以上升到40 MHz。RAM数据保存电压被指定为1.5 V，如果电源电压低于这个门限值，那么数据将会丢失。在实践中，大部分基于微控制器的系统都工作在一个由电压调节器提供的单一+5 V电源下。

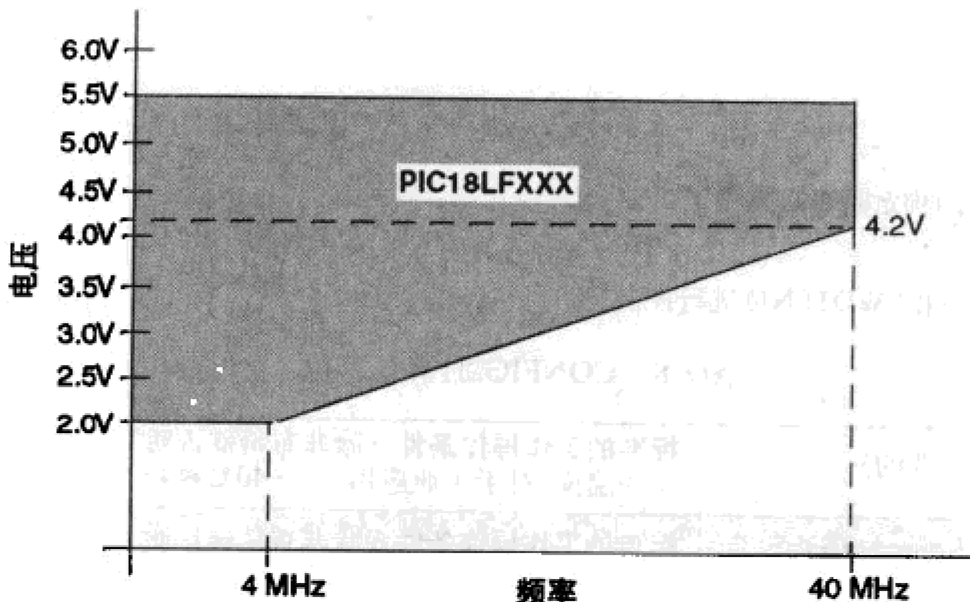


图2-10　PIC18LF452微控制器在不同电压下的工作频率

2.1.5　复位

57

复位操作可以将微控制器置于一个已知的状态。复位PIC18F微控制器，将从程序存储器的0000H地址开始执行程序。复位微控制器包括以下几种类型：

- 上电复位（POR）
- MCLR复位
- 看门狗定时器（WDT）复位
- 掉电复位（BOR）
- 复位指令
- 栈满复位
- 栈下溢复位

58

常用的两种复位类型是上电复位和使用MCLR引脚触发的外部复位。

上电复位

当向微控制器施加供电电压时，上电复位将会自动进行。MCLR引脚应该直接连接到工作电源上或者串联一个10 kΩ的电阻器更佳。图2-11所示的是一个常见的复位电路。

对于电压上升时间较慢的实际应用，建议使用一个二极管、一个电容和一个串联的电阻，如图2-12所示。

在有些应用中，必须使用按钮来外部复位微控制器。图2-13给出了一种用于微控制器外部复位的电路。在正常情况下，MCLR引脚的输入是逻辑1。当按下RESET（复位）按钮时，该引脚变成逻辑0，并使得微控制器复位。

59

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

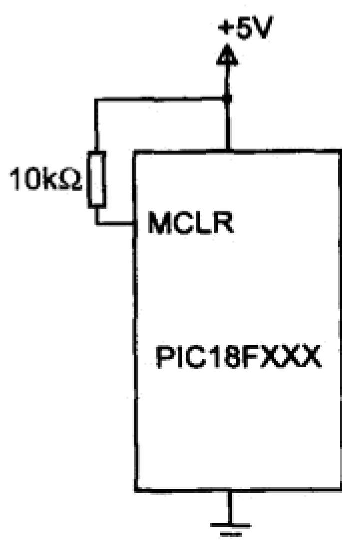


图2-11 典型的复位电路

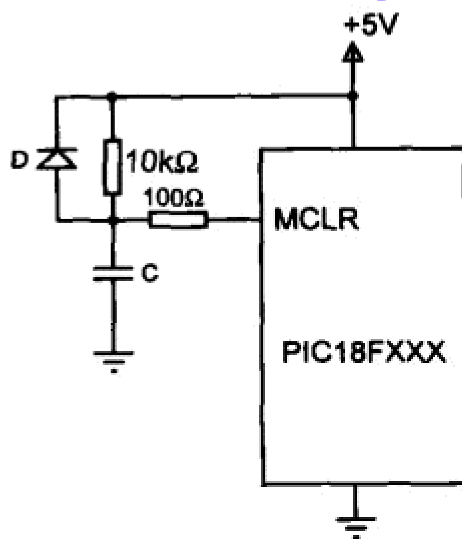


图2-12 电压缓升的复位电路

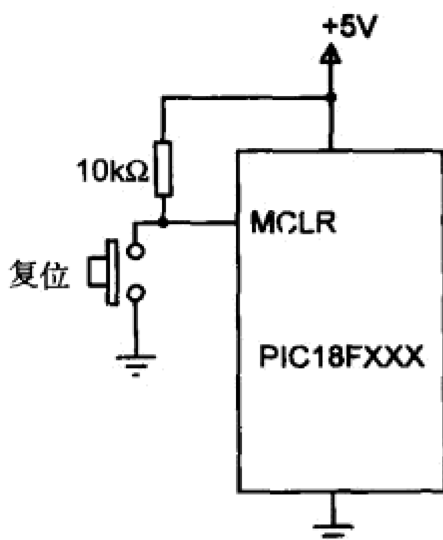


图2-13 外部复位电路

2.1.6 时钟源

PIC18F452微控制器通过一个连接在微控制器OSC1和OSC2引脚上的外部的晶体或陶瓷谐振器来工作。此外，外部的电阻和电容、外部时钟源，以及有些模型的内部振荡器也可以用来为微控制器提供时钟脉冲。PIC18F452微控制器有8个时钟源，可由配置寄存器CONFIG1H来选择。它们分别是：

- 低功率的晶体（LP）
- 晶体或陶瓷谐振器（XT）
- 高速的晶体或陶瓷谐振器（HS）
- 带有PLL的高速晶体或者陶瓷谐振器（HSPLL）
- 在OSC2引脚上的频率为$F_{OSC/4}$的外部时钟（EC）
- 在OSC2引脚（端口RA6）上的带有I/O的外部时钟（ECIO）
- 在OSC2引脚上的带有$F_{OSC/4}$输出的外部电阻/电容（RC）
- 在OSC2引脚（端口RA6）上的带有I/O的外部电阻/电容（RCIO）

60

1. 晶体或陶瓷谐振器

上面列出来的前几个时钟源都使用了一个连接在引脚OSC1和OSC2引脚上的外部晶体或陶

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

瓷谐振器。对于需要精确时间的应用，这里要用到晶体。如果使用到晶体，就必须选择一个并行谐振的晶体，因为在系统第一次上电时串联谐振的晶体是不会起振的。

图2-14说明了一个晶体是如何连接到微控制器上的。其中，电容的值取决于晶体的模式和所选择的频率。表2-4给出了推荐值。例如，对于4MHz的晶体频率，建议使用15pF的电容。更大容量的电容可以提高振荡器的稳定性，但也增加了启动时间。

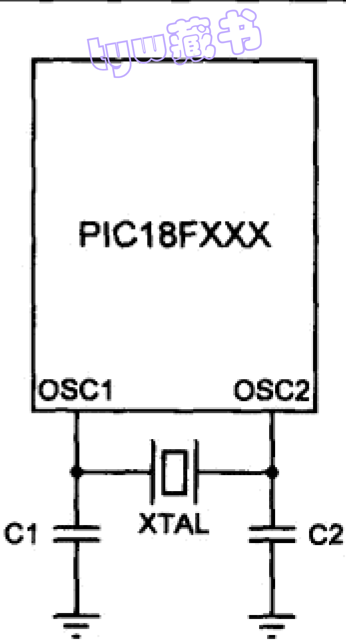


图2-14　使用晶体作为时钟输入

61

谐振器应该在不需要很高时间精度的低成本应用中使用。图2-15说明了一个谐振器是如何连接到微控制器上的。

LP（低功率）振荡器模式适用于时钟频率最大为200 kHz的应用。XT模式建议在时钟频率最大为4 MHz的应用中使用，而HS（高速）模式则建议在时钟频率为4 MHz～25 MHz之间的应用中使用。

表2-4　电容值

模　式	频　率	C1,C2(pF)
LP	32 kHz	33
	200 kHz	15
XT	200 kHz	22～68
	1.0 MHz	15
	4.0 MHz	15
HS	4.0 MHz	15
	8.0 MHz	15～33
	20.0 MHz	15～33
	25.0 MHz	15～33

62

如图2-16所示，在LP、XT或HS模式中，外部时钟源也可以被连接到OSC1引脚。

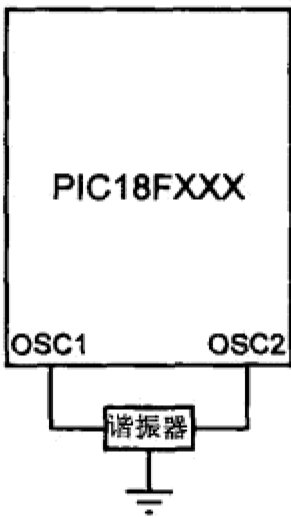


图2-15　使用谐振器作为时钟输入

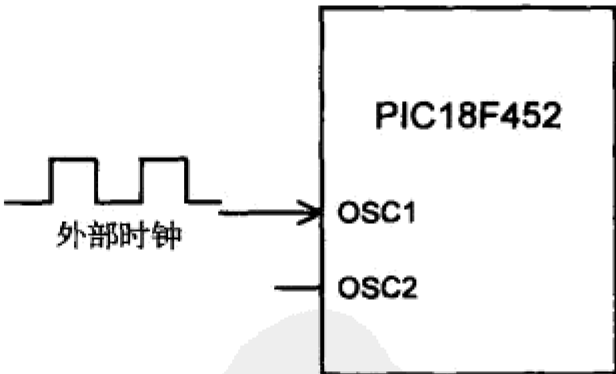


图2-16　在LP、XT或HS模式下连接外部时钟

2. 外部时钟

在EC和ECIO的模式下，外部时钟源可以被连接到微控制器的OSC1引脚输入。在上电复位后，是不需要振荡器启动时间的。图2-17给出了在EC模式下的外部时钟连接。在OSC2引脚上可以得到$F_{OSC/4}$的时序脉冲。这些脉冲可用作测试目的或者用来为外部设备提供脉冲信号。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

除了OSC2引脚可用作通用的数字I/O引脚外，ECIO模式和EC模式很相似。如图2-18所示，该引脚用作了PORTA端口的第6位，即引脚RA6。 63

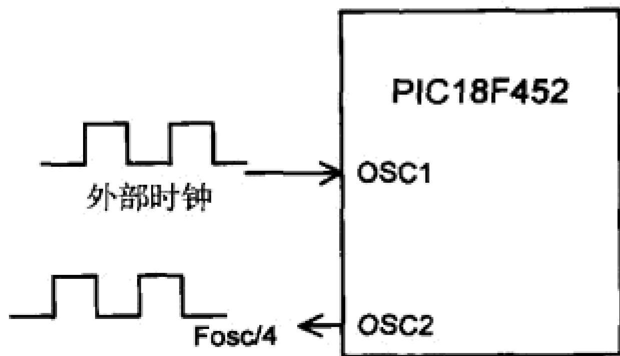


图2-17 在EC模式下的外部时钟

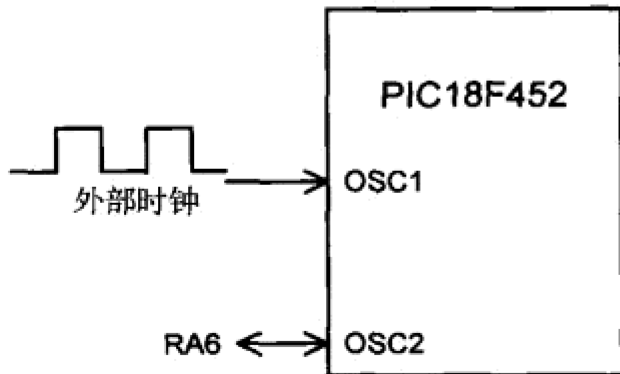


图2-18 在ECIO模式下的外部时钟

3. 电阻/电容

在许多并不需要精确时间的应用中，可以使用一个外部电阻和一个电容来提供时钟脉冲。时钟频率是关于电阻、电容、电源电压和温度的函数。这样得到的时钟频率不是很精确，并且由于制造工艺和元件误差会引起很大的变化。表2-5给出了在不同电阻和电容组合下时钟频率的近似值。当R的值在3 kΩ～100 kΩ之间且C的值大于20 pF时，得到一个非常近似的时钟频率是1/(4.2RC)。

在RC模式下，在微控制器的OSC2引脚上可以得到$F_{OSC/4}$的振荡器频率。图2-19给出了时钟频率近似2MHz的连接，其中R=3.9 kΩ，C=30 pF。在这种情况下，OSC2引脚输出的时钟频率为2 MHz/4=500 kHz。 64

表2-5 不同RC值对应的时钟频率

C(pF)	R(kΩ)	频率（MHz）
22	3.3	3.3
	4.7	2.3
	10	1.08
30	3.3	2.4
	4.7	1.7
	10	0.793

除了OSC2引脚可用作通用的数字I/O引脚外，RCIO模式和RC模式很相似。如图2-20所示，该引脚用作了PORTA端口的第6位，即引脚RA6。

4. 带有PLL的晶体或谐振器

使用高频的晶体或谐振器时面临的问题之一就是电磁干扰。PLL（锁相环）电路可用来使时钟频率乘以4倍。因此，对于一个10 MHz的晶体时钟频率，其内部的工作频率将倍乘为40 MHz。 65当振荡器的配置位被编程为HS模式时，PLL模式有效。

5. 内部时钟

PIC18F系列中有些设备带有内部时钟模式（尽管PIC18F452微控制器没有）。在这种模式下，OSC1和OSC2引脚可用作通用的I/O引脚（RA6和RA7)或者用作$F_{OSC/4}$和RA7。内部时钟的频率范围是31 kHz～8 MHz，由寄存器OSCCON和OSCTUNE来选择。图2-21给出了内部时钟寄存器的位描述。

6. 时钟转换

将时钟从主振荡器转换到一个低频率的时钟源是可能的。例如，可在工作负荷繁忙时使时钟频率更快，而在工作负荷较少时降低时钟频率。在PIC18F452微控制器中，时钟的转换由寄

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

存器OSCCON中的位SCS来控制。在支持内部时钟的PIC18F系列微控制器中，时钟的转换由OSCCON寄存器的位SCS0和SCS1来控制。在时钟转换期间，保证时钟信号不发生故障是非常 66 重要的。PIC18F系列微控制器内置有专门的电路来保证从一个频率转换到另一个频率不出错。

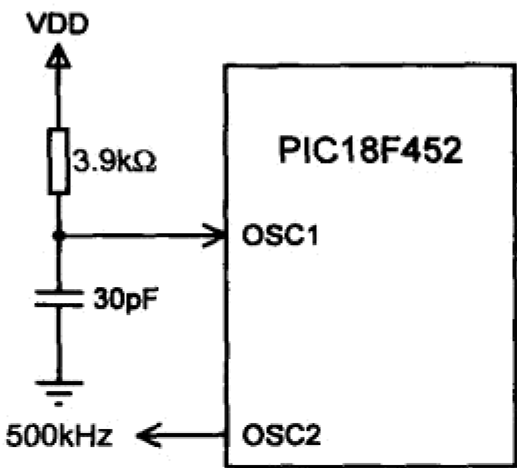


图2-19　在RC模式下的2 MHz时钟

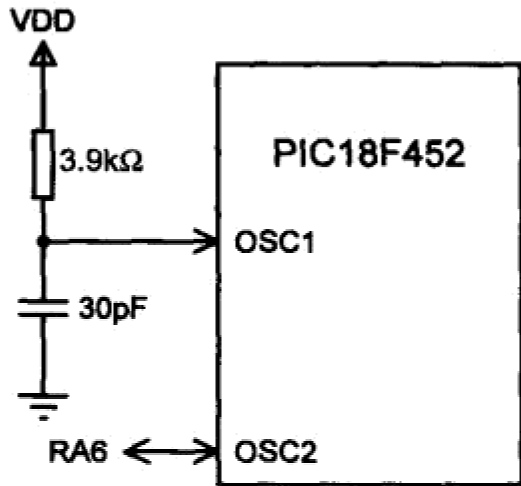


图2-20　在RCIO模式下的2 MHz时钟

OSCCON寄存器

IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCSI	SCS0

IDLEN	0	运行模式有效
	1	空闲模式有效
IRCF2：IRCF0	000	31 kHz
	001	125 kHz
	010	250 kHz
	011	500 kHz
	100	1 MHz
	101	2 MHz
	110	4 MHz
	111	8 MHz
OSTS	0	振荡器启动定时器运行
	1	振荡器启动定时器失效
IOFS	0	内部振荡器不稳定
	1	内部振荡器稳定
SCSI：SCS0	00	主振荡器
	01	定时器1振荡器
	10	内部振荡器
	11	内部振荡器

OSCTUNE寄存器

X	X	T5	T4	T3	T2	T1	T0

XX011111	最大频率
XX000001	
XX000000	中心频率
XX111111	
XX100000	最小频率

图2-21　内部时钟控制寄存器

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

2.1.7 看门狗定时器

在PIC18F系列微控制器中，看门狗定时器（WDT）是一个在芯片上自由运行的基于RC的振荡器，不需要任何的外部器件。当WDT超时的时候，设备就会产生复位（RESET）。如果设备处于睡眠（SLEEP）模式，WDT超时将会唤醒它并继续正常的运行。

看门狗定时器的使能/禁止(enabled/disabled)可由寄存器WDTCON的位SWDTEN来决定。设定SWDTEN=1将使能WDT，而将该位清零则将关闭WDT。在PIC18F452微控制器上，一个8位的后分频器用来将基本的超时周期乘以一个1～128内的值（以2的幂次表示）。这个后分频器的值由配置寄存器CONFIG2H进行控制。后分频器值为1时，典型的基本WDT超时周期是18 ms。 67

2.1.8 并行 I/O 接口

PIC18F系列微控制器的并行端口和PIC16系列的并行端口十分相似。I/O端口和接口引脚的数量取决于所使用的PIC18F微控制器，但是它们中所有微控制器都至少有PORTA和PORTB端口。端口的引脚用RPn表示，其中P表示端口字符，而n表示端口的位。例如，PORTA端口引脚用来表示从RA0到RA7，PORTB端口引脚用来表示从RB0到RB7，等等。

对于一个可工作的端口，可以执行下面的操作：

- 设置端口的方向；
- 设置一个输出值；
- 读取一个输入值；
- 设置一个输出值，然后读其返回值。

在PIC16系列和PIC18F系列中，前三项操作是一样的。在有些应用中，我们可能想传送一个值到端口，然后再读回刚刚传送的值的返回值。PIC16系列微控制器在端口设计上是有缺陷的，以至于从端口读取的值同刚写入端口的值可能会不同。这是因为读取的是实际端口引脚上的值，而这个值可能被连接到该端口引脚上的外部设备改变了。在PIC18F系列中，为了保持发送到端口引脚的实际值不变，将一个锁存寄存器（如给PORTA端口的LATA寄存器）引入到I/O端口。此时，执行读端口操作将会读取锁存的值，而不会受到任何外部设备的影响。

在本节中，我们将介绍I/O端口的通用结构。

1. PORTA

在PIC18F452微控制器中，PORTA是7位宽的，并且端口引脚还具有其他的功能。表2-6给出了PORTA端口引脚的功能。 68

表2-6 PIC18F452的PORTA端口引脚的功能

引　脚	描　述	引　脚	描　述
RA0/AN0		AN3	模拟输入3
RA0	数字I/O	VREF+	A/D参照电压（高）输入
AN0	模拟输入0	**RA4/TOCKI**	
RA1/AN1		RA4	数字I/O
RA1	数字I/O	T0CKI	定时器0外部时钟输入
AN1	模拟输入1	**RA5/AN4/SS/LVDIN**	
RA2/AN2/VREF-		RA5	数字I/O
RA2	数字I/O	AN4	模拟输入4
AN2	模拟输入2	SS	SPI从选择输入
VREF-	A/D参照电压（低）输入	**RA6**	数字I/O
RA3/AN3/VREF+			
RA3	数字I/O		

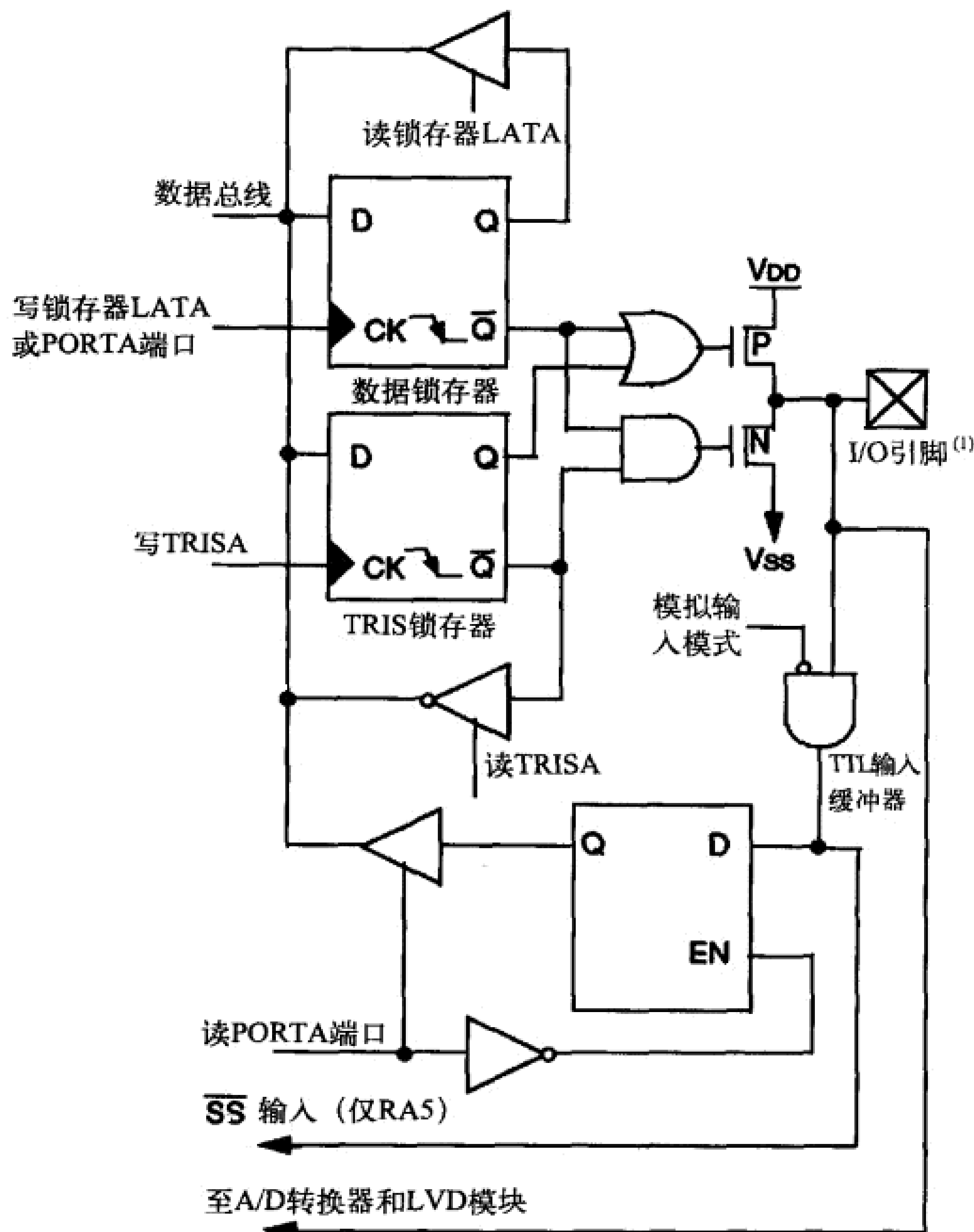
69

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

PORTA端口的结构如图2-22所示。同PORTA端口有关的3个寄存器是：

- 端口数据寄存器—PORTA
- 端口数据方向寄存器—TRISA
- 端口锁存寄存器—LATA



注解1：I/O引脚都有连接到V_{DD}和V_{SS}的保护二极管。

70

图2-22　PIC18F452的PORTA端口的RA0～RA3引脚和RA5引脚

PORTA是端口数据寄存器的名字。TRISA寄存器定义了PORTA引脚的方向：当TRISA的某位为逻辑1时，则对应的引脚被定义为输入引脚；而当TRISA的某位为逻辑0时，则对应的引脚被定义为输出引脚。LATA是一个输出锁存寄存器，与PORTA端口共享相同的锁存器。将数据写入其中一个寄存器，等效于向另外一个寄存器也写入相同的数据。但是，读取LATA寄存器将激活图2-22顶部的缓冲器，保存在PORTA/LATA数据锁存器中的值将被传输到数据总线上，不管微控制器的实际输出引脚的状态如何。

PORTA的第0位到第3位以及第5位也可用作模拟输入。在设备复位后，这些引脚被编程为模拟输入，而RA4和RA6被配置为数字输入。为了将模拟输入编程为数字I/O，必须对ADCON1寄存器（A/D寄存器）做相应的编程。向ADCON1寄存器写入7，将所有的PORTA端口引脚配置为数字I/O。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

RA4引脚与定时器0的时钟输入（T0CKI）是复用的。该引脚是一个施密特触发器输入和一个开漏极输出。

RA6引脚也可以用作通用I/O引脚、或者OSC2时钟输入、或者用作时钟输出来提供$F_{OSC/4}$的时钟脉冲。

2. PORTB

在PIC18F452微控制器中，PORTB是一个8位宽的双向端口，与中断引脚和串行设备编程引脚共用。表2-7给出了PORTB端口各位的功能。

表2-7 PIC18F452的PORTB端口引脚的功能

引 脚	描 述	引 脚	描 述
RB0/INT0		CCP2	捕捉2输入，比较2和PWM2输出
RB0	数字I/O	**RB4**	数字I/O，变化引脚上的中断
INT0	外部中断0	**RB5/PGM**	
RB1/INT1		RB5	数字I/O，变化引脚上的中断
RB1	数字I/O	PGM	低压ICSP编程引脚
INT1	外部中断1	**RB6/PGC**	
RB2/INT2		RB6	数字I/O，变化引脚上的中断
RB2	数字I/O	PGC	内电路调试器和ICSP编程引脚
INT2	外部中断2	**RB7/PGD**	
RB3/CCP2		RB7	数字I/O，变化引脚上的中断
RB3	数字I/O	PGD	内电路调试器和ICSP编程引脚

PORTB端口受以下的三个寄存器所控制：

- 端口数据寄存器—PORTB；
- 端口方向寄存器—TRISB；
- 端口锁存寄存器—LATB。

PORTB的一般操作和PORTA是很相似的。图2-23给出了PORTB端口的结构。每个端口引脚都有一个弱内部上拉电阻，清零INTCON2寄存器的RBPU位将会使其有效。在上电复位期间和将端口引脚配置为输出时，这些弱内部上拉电阻是无效的。在上电复位的时候，PORTB引脚被配置为数字输入引脚。内部的上拉电阻允许像开关这样的器件在不使用外部上拉电阻的情况下就可以连接到PORTB引脚上。这样就减少了元件数量和线路需求，从而节约了成本。 71~72

端口引脚RB4-RB7可用作变化中断输入引脚。当引脚4到引脚7中任意一个引脚上有变化时，都会引起一个中断标志被置位。中断使能位、标志位RBIE和RBIF都位于寄存器INTCON中。

3. PORTC、PORTD、PORTE和更多

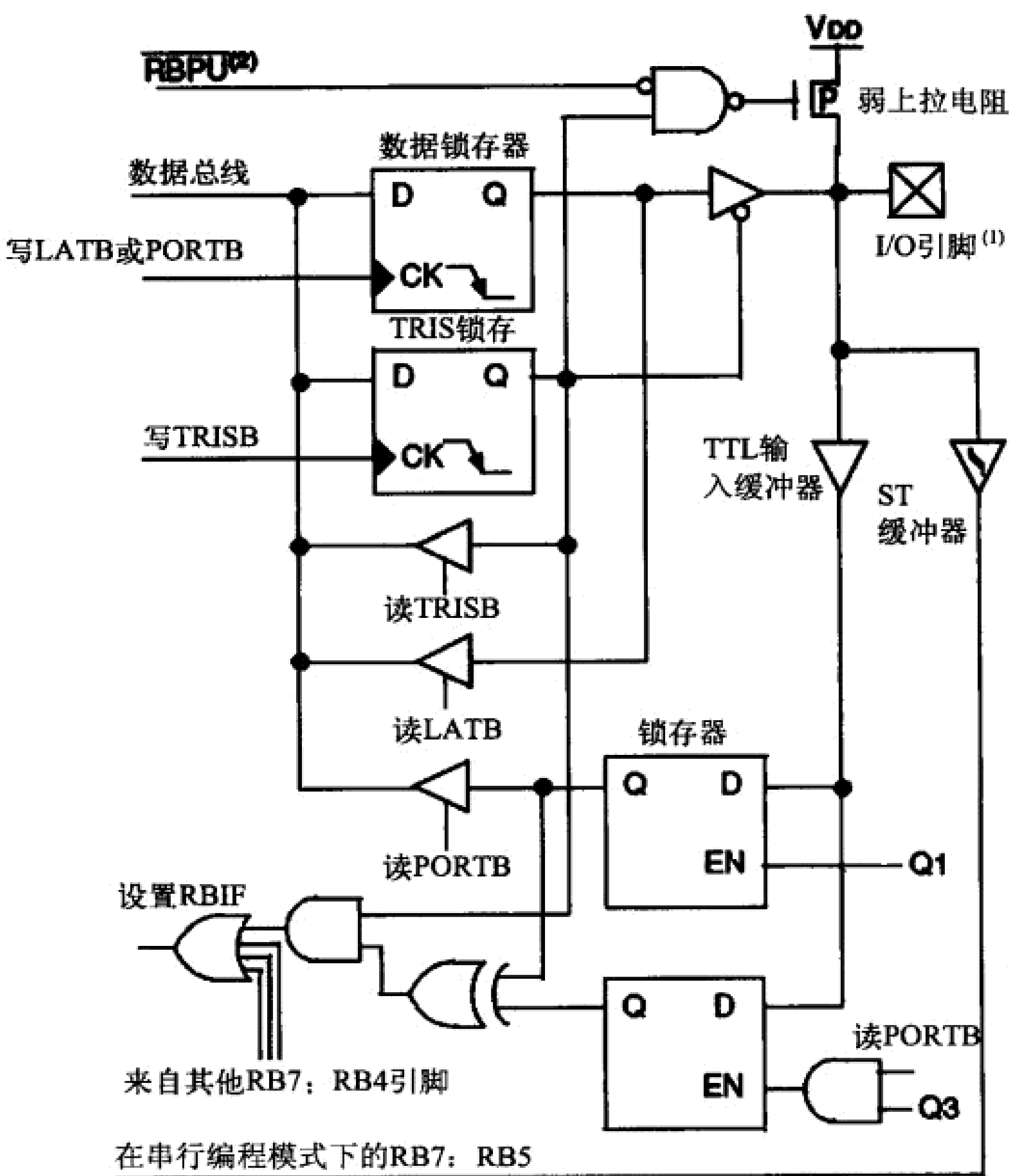
除了PORTA和PORTB以外，PIC18F452微控制器还带有两个8位的双向端口PORTC和PORTD，以及一个3位的端口PORTE。每个端口都有自己的数据寄存器（如PORTC寄存器）、数据方向寄存器（如TRISC寄存器）和数据锁存寄存器（如LATC寄存器）。这些端口的一般操作跟PORTA端口的操作很相似。 73

在PIC18F452微控制器中，PORTC端口引脚同几个外围模块功能是复用的，如表2-8所示。在上电复位时，PORTC引脚被配置为数字输入。

在PIC18F452微控制器中，PORTD端口带有施密特触发器输入的缓冲器。在上电复位时，PORTD被配置为数字输入。通过设置TRISE寄存器的第4位，可以将PORTD配置为一个8位的并行从端口（如一个微处理器端口）。表2-9给出了PORTD端口引脚的功能描述。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书



注解1：I/O引脚都有连接到V_{DD}和V_{SS}的保护二极管。

2：为了使能弱上拉电路，需把TRIS相应的位设置为1，并且清零RBPU位（INTCON2<7>）。

图2-23　PIC18F452的PORTB端口的RB4～RB7引脚

表2-8　PIC18F452的PORTC端口引脚的功能描述

引　脚	描　述	引　脚	描　述
RC0/T1OSO/T1CKI		**RC4/SDI/SDA**	
RC0	数字I/O	RC4	数字I/O
T1OSO	定时器1振荡器输出	SDI	SPI数据输入
T1CKI	定时器1/定时器3外部时钟输入	SDA	IC数据I/O
RC1/T1OSI/CCP2		**RC5/SDO**	
RC1	数字I/O	RC5	数字I/O
T1OSI	定时器1振荡器输入	SDO	SPI数据输出
CCP2	捕捉2输入、比较2和PWM2输出	**RC6/TX/CK**	
RC2/CCP1		RC6	数字I/O
RC2	数字I/O	TX	USART发送引脚
CCP1	捕捉1输入、比较1和PWM1输出	CK	USART同步时钟引脚
RC3/SCK/SCL		**RC7/RX/DT**	
RC3	数字I/O	RC7	数字I/O
SCK	对于SPI的同步串行时钟输入/输出	RX	USART接收引脚
SCL	对于IC的同步串行时钟输入/输出	DT	USART同步数据引脚

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

表2-9 PIC18F452的PORTD端口引脚的功能描述

引 脚	描 述	引 脚	描 述
RD0/PSP0		**RD4/PSP4**	
RD0	数字I/O	RD4	数字I/O
PSP0	并行从端口位0	PSP4	并行从端口位4
RD1/PSP1		**RD5/PSP5**	
RD1	数字I/O	RD5	数字I/O
PSP1	并行从端口位1	PSP5	并行从端口位5
RD2/PSP2		**RD6/PSP6**	
RD2	数字I/O	RD6	数字I/O
PSP2	并行从端口位2	PSP6	并行从端口位6
RD3/PSP3		**RD7/PSP7**	
RD3	数字I/O	RD7	数字I/O
PSP3	并行从端口位3	PSP7	并行从端口位7

在PIC18F452微控制器中，PORTE只有3位宽。如表2-10所示，这些端口引脚同模拟输入和并行从端口读/写控制位是共享的。在上电复位时，PORTE引脚被配置为模拟输入，因此必须对寄存器ADCON1进行编程，来将这些端口配置为数字I/O。

表2-10 PIC18F452的PORTE端口引脚的功能描述

引 脚	描 述	引 脚	描 述
RE0/RD/AN5		WR	并行从端口读控制引脚
REO	数字I/O	AN6	模拟输入6
RD	并行从端口读控制引脚	**RE2/CS/AN7**	
AN5	模拟输入5	RE2	数字I/O
RE1/WR/AN6		CS	并行从端口CS
RE1	数字I/O	AN7	模拟输入7

2.1.9 定时器

PIC18F452微控制器具有4个可编程的定时器，用于产生时序信号、在指定时间间隔里产生中断、测量频率和时间间隔等许多任务。

本节将介绍PIC18F452微控制器可用的定时器。

1. 定时器0

定时器0和PIC16系列的定时器0很相似，除了能工作在8位或者16位模式下之外。定时器0具有以下的基本特性：

- 8位或者16位操作
- 8位可编程的预分频器
- 外部或内部时钟源
- 溢出时可产生中断

如图2-24所示，定时器0的控制寄存器为T0CON。该寄存器的低6位和PIC16系列的OPTION寄存器有着相似的功能。最高的两位用来选择操作模式是8位还是16位，以及使能/禁止定时器0。

定时器0可以用作定时器或计数器。只要将T0CS位清零就选择了定时器模式，在这种模式

74
~
76

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

下定时器的时钟来自$F_{OSC/4}$。将T0CS置位就选择了计数器模式，在这种模式下定时器0在输入引脚RA4/T0CKI的上升沿或下降沿进行增量计数。T0CON寄存器的T0SE位用来选择边沿触发模式。

77

一个8位的预分频器用来改变定时器的时钟频率，其取值最大为256。该预分频器值由寄存器T0CON的PSA位和T0PS2:T0PS0位进行选择。

8位模式　图2-25给出了8位模式下的定时器0。在定时器应用中通常使用到下面的操作：

- 将T0CS清零以选择时钟$F_{osc/4}$
- 使用T0PS2:T0PS0位来选择一个合适的预分频器值
- 将PSA清零来选择预分频器
- 加载定时器寄存器TMR0L
- 允许定时器中断（可选）
- 定时器开始计数，当在8位模式下定时器的计数值从FFH到00H（在16位模式下从FFFFH到0000H）溢出时，产生中断。

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
位7							位0

位7　TMR0ON：定时器0的开/关控制位
1=定时器0开启
0=定时器0停止

位6　T08BIT：定时器0的8位/16位控制位
1=将定时器配置为一个8位定时器/计数器
0=将定时器配置为一个16位定时器/计数器

位5　T0CS：定时器0的时钟源选择位
1=转换到T0CK1引脚
0=内部指令周期时钟（CLKO）

位4　T0SE：定时器0的时钟边沿选择位
1=在T0CKI引脚信号从高电平转换到低电平时增量计数
0=在T0CKI引脚信号从低电平转换到高电平时增量计数

位3　PSA：定时器0的预分频器指定位
1=定时器0的预分频器未指定。定时器0时钟输入忽略预分频器
0=定时器0的预分频器已被指定，定时器0时钟输入来自预分频器的输出

位2~0　T0PS2:T0PS0：定时器0预分频器选择位
111=1:256预分频器值
110=1:128预分频器值
101=1:64预分频器值
100=1:32预分频器值
011=1:16预分频器值
010=1:8预分频器值
001=1:4预分频器值
000=1:2预分频器值

图2-24　定时器0的控制寄存器（T0CON）

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

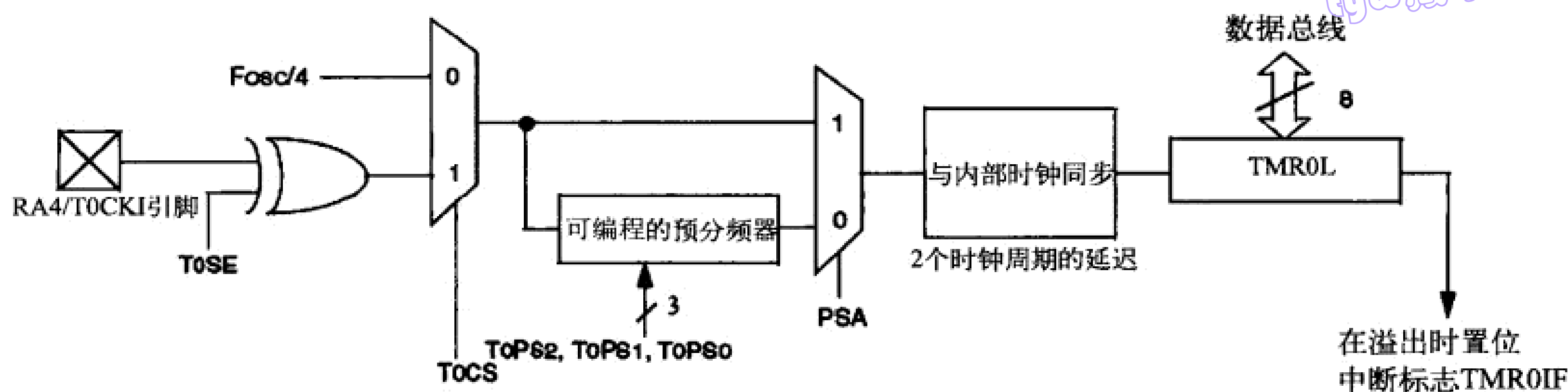


图2-25 在8位模式下的定时器0

通过向TMR0寄存器载入一个初值，可以控制定时器0的计数直到产生溢出。下面的公式可用来计算在给定振荡器周期、定时器初值和预分频器值的情况下产生溢出（或发生中断）所需的时间：

$$溢出时间=4 \times T_{osc} \times 预分频器值 \times (256-TMR0) \quad (2.1)$$

其中，

溢出时间的度量单位是μs

T_{osc}是振荡周期，度量单位是μs

预分频器值是指预分频器的值

TMR0是指载入到TMR0寄存器的初值

例如，假设使用频率为4MHz的晶体，通过将PS2:PS0位设置为010来选择1:8的预分频器。再假设载入到定时器寄存器TMR0的初值是十进制数100。那么溢出时间为：

$$4MHz时钟的周期为T=1/f=0.25\ \mu s$$

使用上面的公式，可以得到：

$$溢出时间=4 \times 0.25 \times 8 \times (256-100)=1\ 248\ \mu s$$

因此，定时器将在1.248ms后发生溢出。如果使能定时器中断和全局中断，那么还会发生定时器中断。

通常，对于一个期望的溢出时间，需要指定一个载入到TMR0寄存器的初值。这可以通过修改式（2.1）来计算如下：

78
~
79

$$TMR0=256-(溢出时间)/(4 \times T_{osc} \times 预分频器) \quad (2.2)$$

例如，假设希望在500 μs后产生中断，所使用的时钟和预分频器的值同前面一样。那么使用式（2.2）可计算出载入TMR0寄存器的初值如下：

$$TMR0=256-500/(4 \times 0.25 \times 8)=193.5$$

因此，载入到TMR0寄存器的最接近的数为193。

16位模式 在16位模式下的定时器0，如图2-26所示。这里，使用两个定时器寄存器TMR0L和TMR0来存储16位的定时器初值。低字节的TMR0L寄存器直接通过数据总线载入。高字节的TMR0寄存器可以通过一个叫作TMR0H的缓冲器载入。在读TMR0L寄存器期间，定时器（TMR0）的高字节也可以载入到TMR0H，这样所有16位的定时器值都能被读取到。为了读取16位的定时器值，必须先读取TMR0L，然后在紧跟的指令中读取TMR0H。相似地，在写TMR0L寄存器期间，定时器的高字节同样需要使用TMR0H的内容来更新，允许将所有16位的值写入定时器。因此，为了能写入定时器，程序首先应该把期望的高字节写入TMR0H。当低字节被写入TMR0L后，存储在TMR0H中的值就会自动地转移到TMR0，从而将16位写入定时器。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

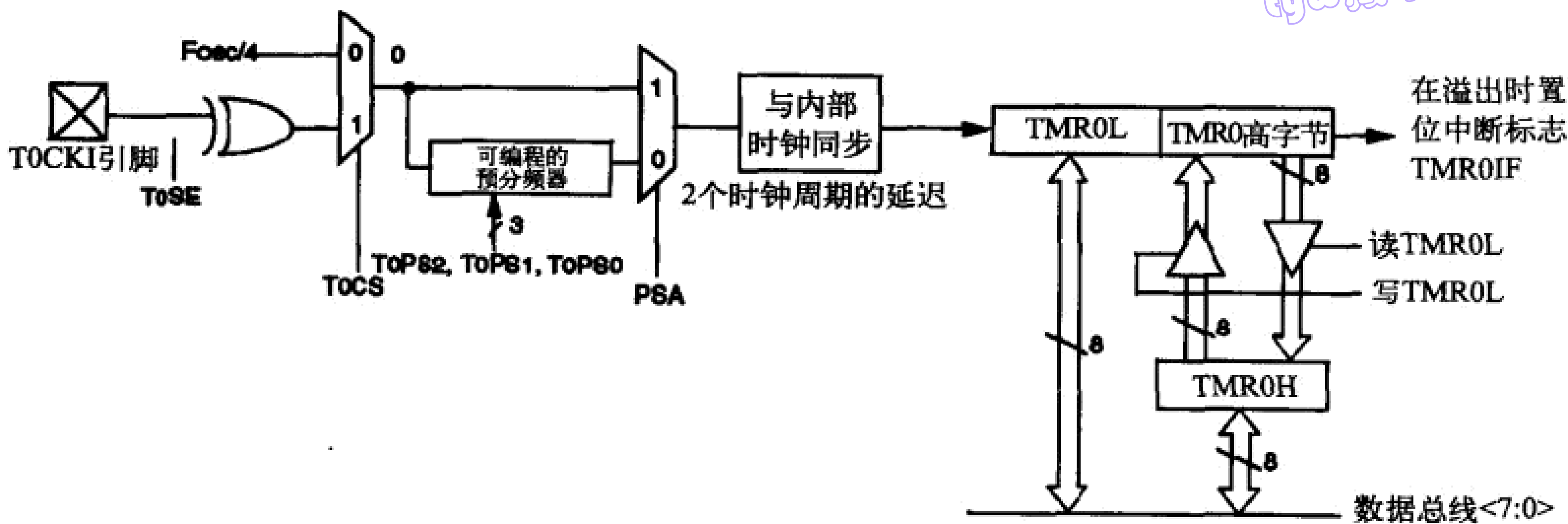


图2-26　在16位模式下的定时器0

2. 定时器1

如图2-27所示，PIC18F452的定时器1是一个由寄存器T1CON控制的16位定时器。图2-28描述了定时器1的内部结构。

80

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
位7							位0

位7　RD16：16位读/写模式使能位
1=使能定时器1的寄存器读/写操作以16位进行
0=使能定时器1的寄存器读/写操作以8位进行

位6　未实现：读出为“0”

位5～4　T1CKPS1:T1CKPS0：定时器1的输入时钟预分频器选择位
11=1：8 预分频器值
10=1：4 预分频器值
01=1：2 预分频器值
00=1：1 预分频器值

位3　T1OSCEN：定时器1的振荡器使能位
1=定时器1的振荡器开启
0=定时器1的振荡器关闭
关闭振荡器反相器和反馈电阻以可降低功率消耗。

位2　$\overline{\text{T1SYNC}}$：定时器外部时钟输入同步选择位
当TMR1CS=1时：
1=外部时钟不同步
0=外部时钟同步
当TMR1CS=0时：
这个位被忽略，定时器1使用内部时钟。

位1　TMR1CS：定时器1的时钟源选择位
1=来自引脚RC0/T1OSO/T13CKI的外部时钟（在上升沿）
0=内部时钟（$F_{OSC/4}$）

位0　TMR1ON：定时器1的使能位
1=打开定时器1
0=关闭定时器1

图2-27　定时器1的控制寄存器（T1CON）

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

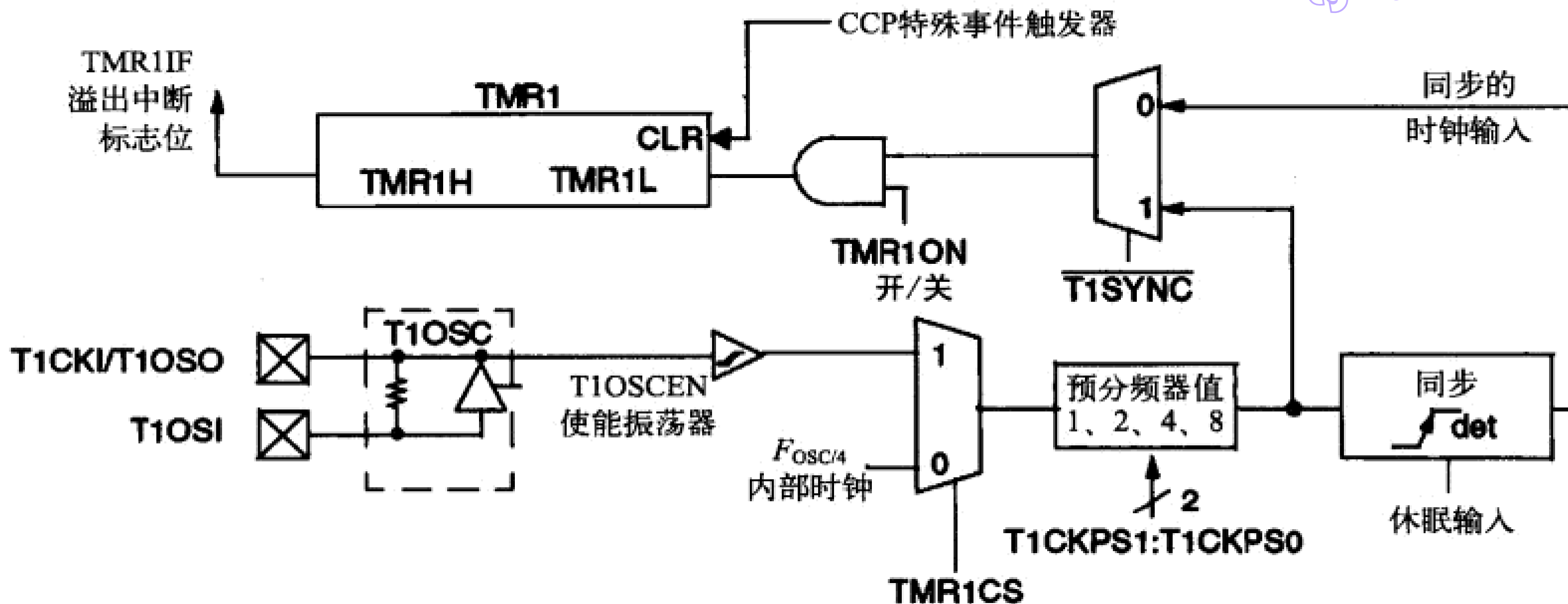


图2-28 定时器1的内部结构

定时器1可用作定时器或计数器。当寄存器T1CON的TMR1CS位为低电平时，时钟$F_{OSC/4}$被选择为定时器的输入。当TMR1CS位为高电平时，定时器1用作T1OSI输入的计数器。由T1CON寄存器的T1OSCEN位控制的晶体振荡器电路，连接在T1OSI和T1OSO引脚之间，其中支持的晶体可达到200 kHz。这个振荡器在实时时钟应用中主要使用32 kHz的晶体。在定时器1中使用预分频器来改变时序信号频率，其取值可以是1、2、4或8。

81

对定时器1进行配置，可实现16位操作模式或者2个8位的操作模式。寄存器T1CON的位RD16控制着模式的选择。当RD16是低电平时，定时器的读和写操作都可以看作两个8位的操作。当RD16是高电平时，定时器的读和写操作跟定时器0的16位模式一样（即在定时器寄存器和数据总线之间使用了一个缓冲器）（如图2-29所示）。

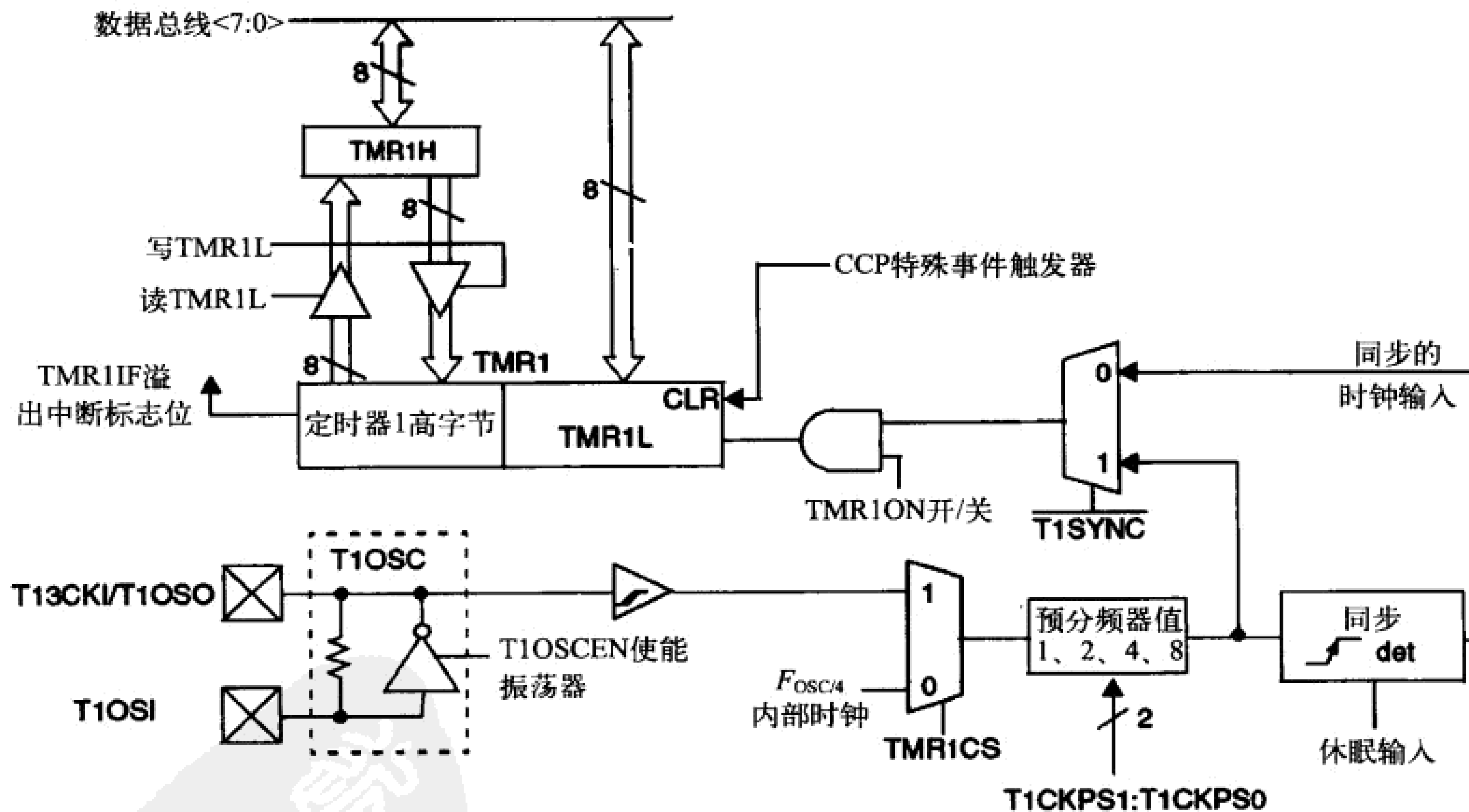


图2-29 在16位模式下的定时器1

如果定时器1中断是允许的，当定时器的值从FFFFH跳变到0000H的时候，将会发生中断。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

3. 定时器2

定时器2是一个8位定时器，具有以下的特性：

- 8位的定时器（TMR2）；
- 8位的周期寄存器（PR2）；
- 可编程的预分频器；
- 可编程的后分频器；
- 当TM2和PR2匹配时，产生中断。

82~83

如图2-30所示，定时器2是由寄存器T2CON控制的。T2CKPS1:T2CKPS0位用来将预分频器值设为1、4和16。TOUTPS3:TOUTPS0位用来将后分频器值设为从1:1到1:16。可以通过对TMR2ON位的置1或者清零来控制定时器的打开或者关闭。

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
位7							位0

位7　未实现：读出为“0”

位6~3　TOUTPS3:TOUTPS0：定时器2的输出后分频器选择位
0000=1:1后分频器
0001=1:2后分频器
•
•
•
1111=1:16后分频器

位2　TMR2ON：定时器2的使能位
1=定时器开启
0=定时器关闭

位1~0　T2CKPS1:T2CKPS0：定时器2时钟预分频器选择位
00=预分频器值为1
01=预分频器值为4
1X=预分频器值为16

图2-30　定时器2的控制寄存器（T2CON）

定时器2的方框图如图2-31所示。定时器2能用于CCP模块的PWM模式。定时器2的输出可由SSP模块选择作为一个波特率时钟。定时器2从00H开始增加直到与PR2相匹配，从而置位中断标志。然后在下一个循环中将复位为00H。

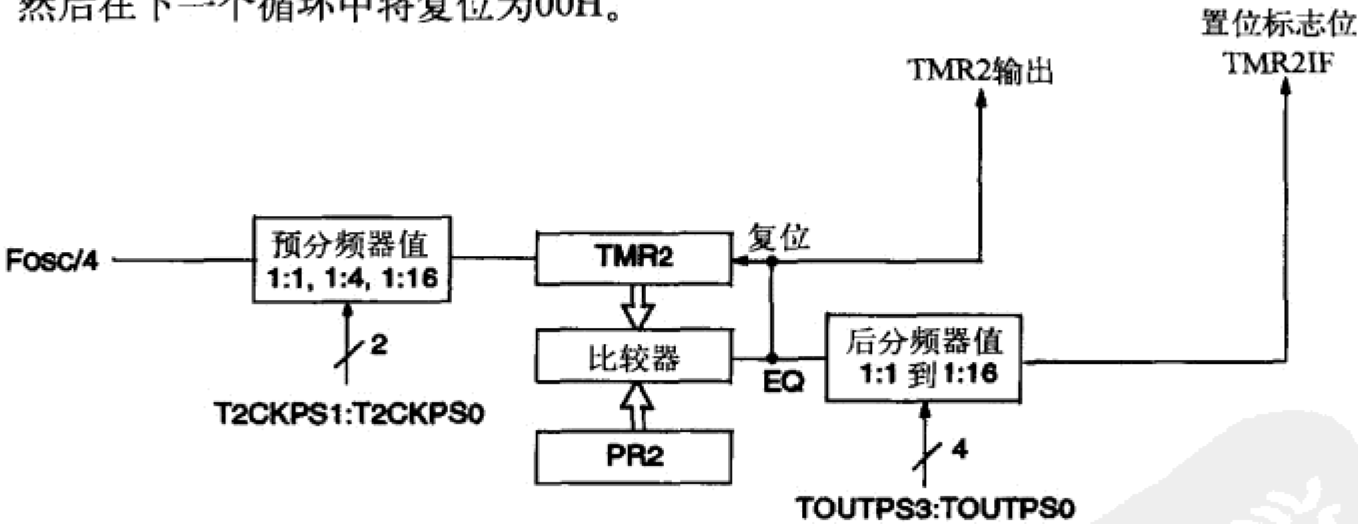


图2-31　定时器2的方框图

4. 定时器3

定时器3的结构和操作跟定时器1是一样的，它有两个寄存器TMR3H和TMR3L。如图2-32

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

所示，该定时器由寄存器T3CON控制。

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
位7							位0

位7　RD16：16位读/写模式使能位
1=使能定时器3寄存器的读/写操作以16位进行
0=使能定时器3寄存器的读/写操作以8位进行

位6~3　T3CCP2:T3CCP1：定时器3和定时器1的CCPx使能位
1x=定时器3是CCP模块的比较/捕捉时钟源
01=定时器3是CCP2的比较/捕捉时钟源，
定时器1是CCP1的比较/捕捉时钟源
00=定时器1是CCP模块的比较/捕捉时钟源

位5~4　T3CKPS1:T3CKPS0：定时器3输入时钟预分频器选择位
11=1:8 预分频器值
10=1:4 预分频器值
01=1:2 预分频器值
00=1:1 预分频器值

位2　$\overline{\text{T3SYNC}}$：定时器3外部时钟输入同步控制位
（如果系统时钟来自定时器1/定时器3，则该位将不可用）
当TMR3CS=1时：
1=外部时钟不同步
0=外部时钟同步
当TMR3CS=0：
这个位被忽略，定时器3使用内部时钟。

位1　TMR3CS：定时器3时钟源选择位
1=来自定时器1振荡器或者T1CKI的外部时钟输入
（在第一个下降沿后的上升沿）
0=内部时钟（$F_{OSC/4}$）

位0　TMR3ON：定时器3的使能位
1=打开定时器3
0=关闭定时器3

图2-32　定时器3的控制寄存器（T3CON）

定时器3的方框图如图2-33所示。

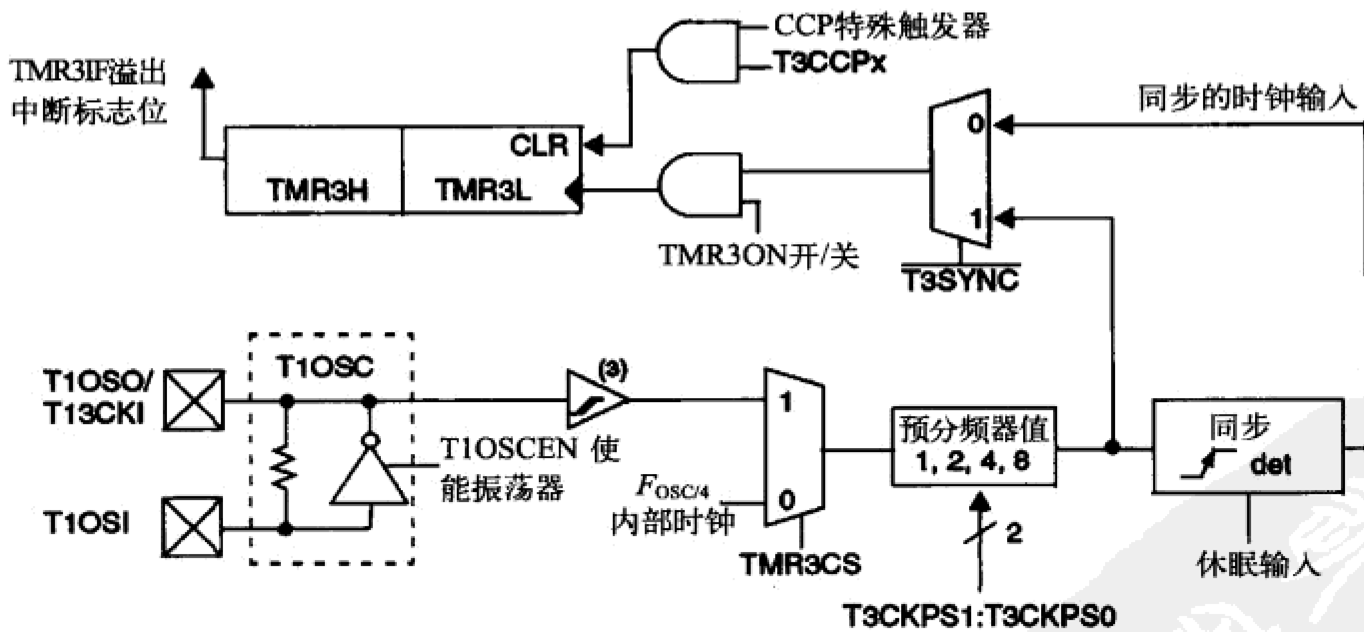


图2-33　定时器3的方框图

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

2.1.10 捕捉/比较/PWM模块（CCP）

PIC18F452微控制器有两个捕捉/比较/PWM（CCP）模块，使用定时器1、定时器2和定时器3来提供捕捉、比较和脉宽调制（PWM）操作。每个模块都有2个8位的寄存器。模块1的寄存器为CCPR1L和CCPR1H，而模块2的寄存器为CCPR2L和CCPR2H。总的来说，每对寄存器组成了一个16位的寄存器，可用来捕捉、比较或者产生指定占空比(duty cycle)的波形。模块1由寄存器CCP1CON控制，而模块2由CCP2CON控制。图2-34给出了CCP控制寄存器的位描述。

84

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
位7							位0

位7~6 未实现：读出为“0”

位5~4 DCxB1:DCxB0：PWM占空比周期位1和位0

捕捉模式：

未使用

比较模式：

未使用

PWM模式：

这些位是10位PWM占空比周期的两个最低有效位（位1和位0）。该占空比周期的高8位（DCx9:DCx2）位于CCPRxL中。

位3~0 CCPxM3：CCPxM0：CCPx模式选择位

0000=捕捉/比较/PWM被禁止（复位CCPx模块）

0001=保留

0010=比较模式，在匹配时切换输出（CCPxIF位被置位）

0011=保留

0100=捕捉模式，每一个下降沿

0101=捕捉模式，每一个上升沿

0110=捕捉模式，每4个上升沿

0111=捕捉模式，每16个上升沿

1000=比较模式

初始化CCP引脚为低电平，在比较匹配时，强制CCP引脚为高电平（CCPIF位被置位）

1001=比较模式

初始化CCP引脚为高电平，在比较匹配时，强制CCP引脚为低电平（CCPIF位被置位）

1010=比较模式

在比较匹配时，产生软件中断（CCPIF位被置位，CCP引脚不受影响）

1011=比较模式

触发特殊事件（CCPIF位被置位）

11xx=PWM模式

图2-34 CCPxCON寄存器的位描述

1. 捕捉模式

在捕捉模式中，寄存器操作起来就像一个秒表。当一个事件发生，该事件的发生时间就会被记录下来，尽管时钟继续在运行（另一方面，当事件时间被记录时，秒表就会停止）。

85

图2-35给出的是捕捉模式的操作。这里将考虑CCP1，CCP2的操作及其寄存器和端口名称可作相应的变化。在这种模式下，当一个事件发生在引脚RC2/CCP1上时（必须使用TRISC将引脚RC2/CCP1配置为输入引脚），CCPR1H:CCPR1L将捕捉TMR1或TMR3寄存器的16位值。外部信号将被4或16预分频。事件由控制位CCP1M3:CCP1M0来选择，可以选择下面的任意事件：

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

- 每一个下降沿；
- 每一个上升沿；
- 每四个上升沿；
- 每十六个上升沿。

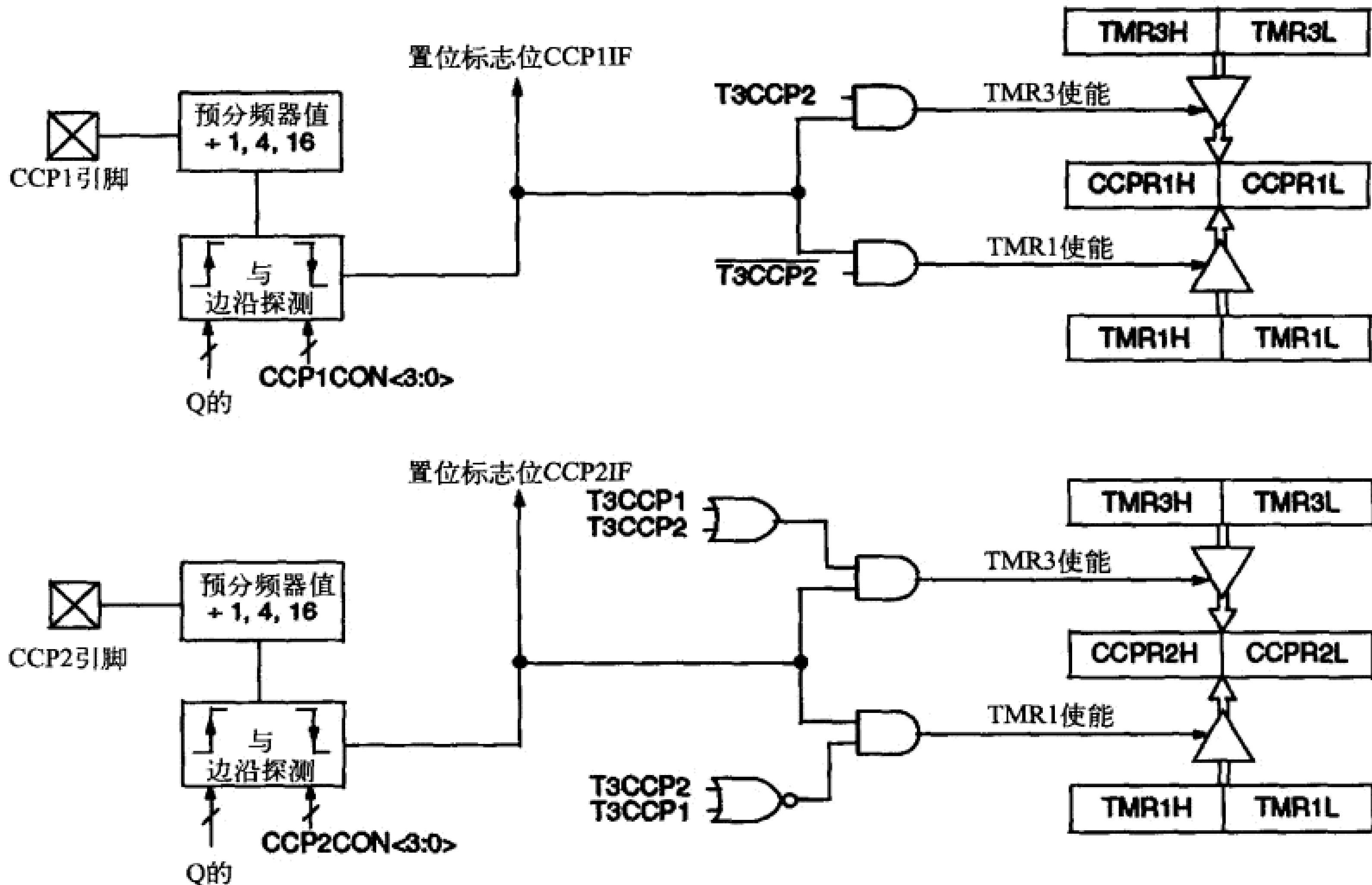


图2-35 捕捉模式

如果使能捕捉中断，那么事件的发生将会在软件上引起中断。如果在读取寄存器CCPR1的值之前出现了另一个捕捉，那么旧的捕捉值就会被新的捕捉值所覆盖。

定时器1和定时器3都可以工作在捕捉模式下。它们必须运行在定时器模式下或者同步的计时器模式，这由寄存器T3CON来进行选择。 86

2. 比较模式

在比较模式中，使用一个数字比较器来比较定时器1或定时器3的值和一对16位寄存器中的值。如果它们相匹配，引脚的输出状态将会发生改变。图2-36给出了比较模式的框图。

这里仅考虑CCP1模块，CCP2模块的操作跟CPP1是一样的。

16位寄存器对（16-bit register pair）CCPR1H:CCPR1L的值是连续地和定时器1或定时器3的值进行比较的。当匹配出现时，RC2/CCP1引脚的状态发生变化，这取决于寄存器CCP1CON的位CCP1M2:CCP1M0的编程设置。下面的状态改变操作是可以编程的： 87

- 强制使得RC2/CCP1为高电平；
- 强制使得RC2/CCP1为低电平；
- 切换RC2/CCP1引脚的状态（从低电平变为高电平或者从高电平变为低电平）；
- 当产生匹配时，产生中断；
- 没有改变。



世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

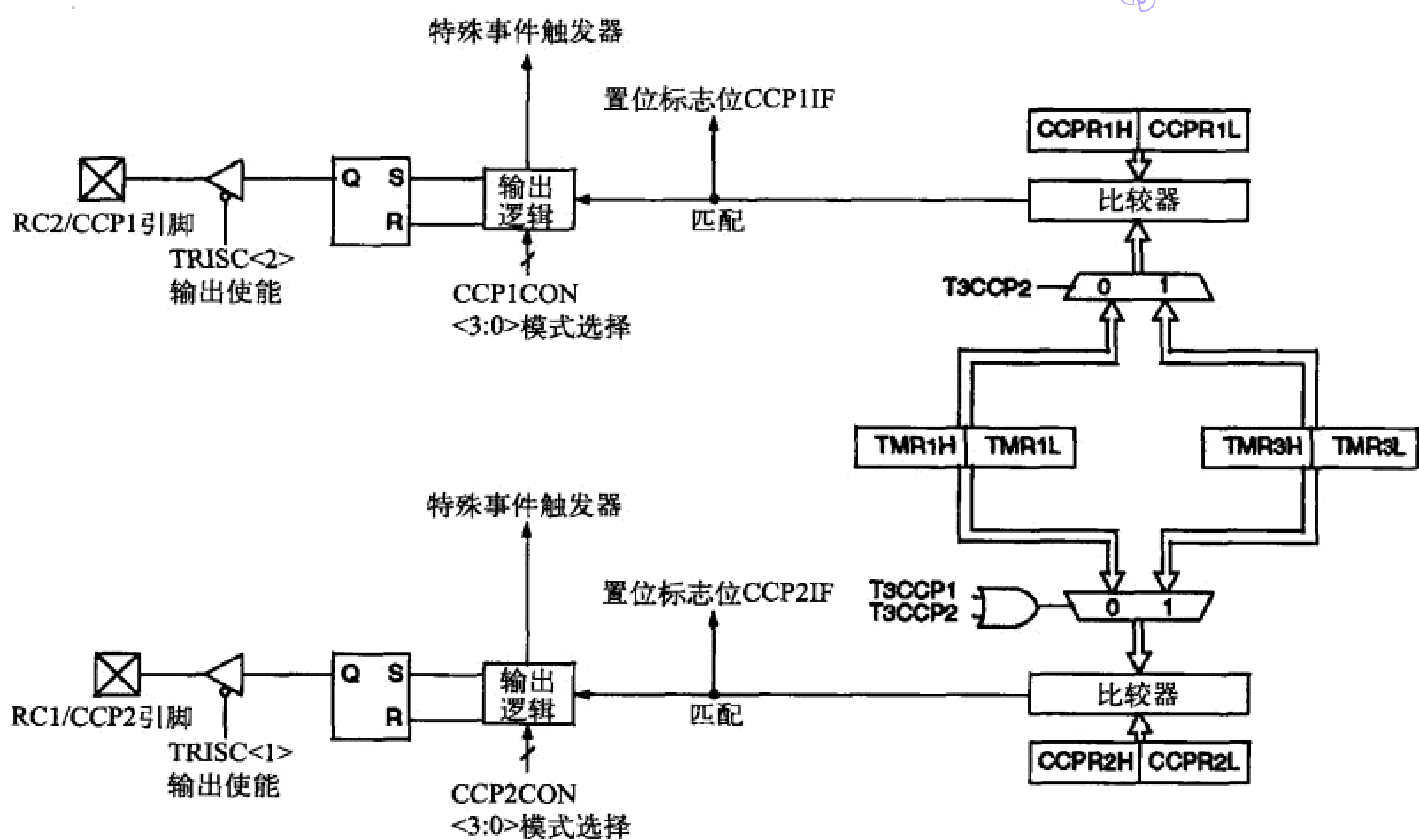


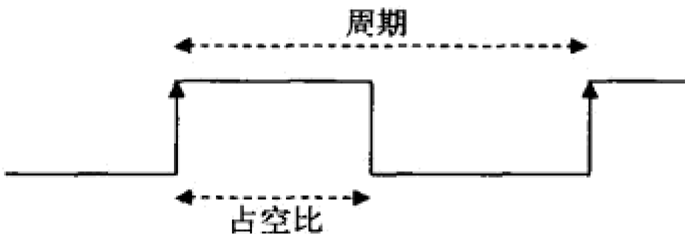
图2-36　比较模式

定时器1或定时器3必须工作在定时器模式或者同步的计数器模式，这由寄存器T3CON来进行选择。

88

3. PWM模块

脉宽调制（PWM）模式可产生10位分辨率的PWM输出。PWM输出基本上是指一个具有指定的周期和占空比的方波。图2-37给出了一个典型的PWM波形。



89

图2-37　典型的PWM波形

图2-38所示的是PWM模块的方框图。该模块由定时器2控制。PWM周期的计算公式为：

$$\text{PWM周期}=(\text{PR2}+1)\times \text{TMR2PS}\times 4\times T_{\text{OSC}} \tag{2.3}$$

或者

$$\text{PR2}=\text{PWM周期}/(\text{TMR2PS}\times 4\times T_{\text{OSC}})-1 \tag{2.4}$$

其中，

PR2是载入到定时器2寄存器中的初值

TMR2PS是定时器2的预分频器值

T_{OSC}是时钟振荡器周期（秒）

PWM频率被定义为1/(PWM周期)

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

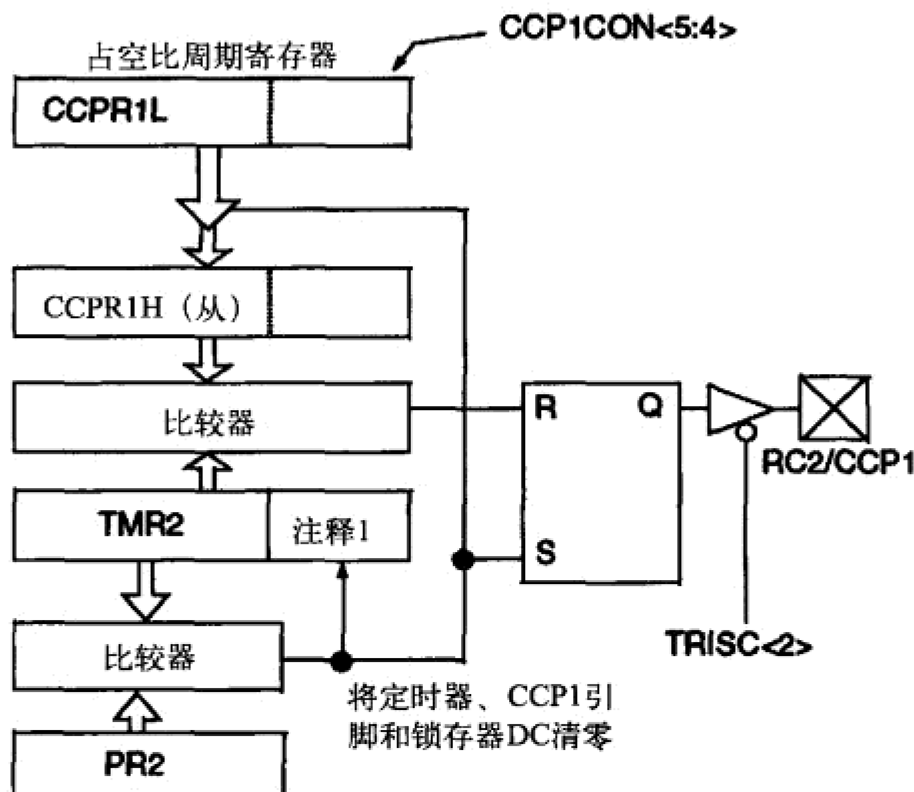


图2-38 PWM模块的方框图

PWM占空比的分辨率为10位。PWM占空比周期由写入到CCPR1L寄存器的8个最高有效位和写入CCP1CON寄存器的位4和位5的两个最低有效位来选择。占空比（单位为秒）的计算公式为： 90

$$\text{PWM占空比周期}=(\text{CCPR1L:CCP1CON}<5:4>)\times \text{TMR2PS}\times T_{OSC} \tag{2.5}$$

或者

$$\text{CCPR1L:CCP1CON}<5:4>=\text{PWM占空比周期}/(\text{TMR2PS}\times T_{OSC}) \tag{2.6}$$

配置PWM模块的操作步骤如下：

- 指定期望的周期和占空比周期；
- 为定时器2的预分频器（TMR2PS）选择一个值；
- 使用式（2.2）计算待写入到PR2寄存器的值；
- 使用式（2.6）计算待载入到CCPR1L和CCP1CON寄存器的值；
- 将TRISC的位2清零，使CCP1引脚配置为输出引脚；
- 使用寄存器CCP1CON，将CCP1模块配置为PWM模式。

下面举例说明如何设置PWM模块。

例2.1 PWM脉冲必须从PIC18F452微控制器的CCP1引脚产生。期望的脉冲周期是44 μs，期望的占空比是50%。假定微控制器使用4MHz的晶体，试计算有关寄存器的初值。

解 使用4MHz晶体，其周期为$T_{OSC}=1/4=0.25\times 10^{-6}$

期望的PWM的占空比周期为44/2=22μs。

假设使用的定时器预分频值为4，根据式（2.4），则有：

$$\text{PR2}=\text{PWM周期}/(\text{TMR2PS}\times 4\times T_{OSC})-1$$

或者 91

$$\text{PR2}=44\times 10^{-6}/(4\times 4\times 0.25\times 10^{-6})-1=10\text{，即0AH}$$

并且根据式（2.6），可以得到

$$\text{CCPR1L:CCP1CON}<5:4>=\text{PWM占空比周期}/(\text{TMR2PS}\times T_{OSC})$$

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

或者

$$CCPR1L:CCP1CON<5:4>=22\times10^{-6}/(4\times0.25\times10^{-6})=22$$

但是若将十进制数22用10位的二进制数表示，则对应于：

"00 00010110"

因此，载入到寄存器CCP1CON的位4和位5的值是"00"。对于PWM模式，寄存器CCP1CON的位2和位3必须被设置为高电平。因此CCP1CON必须被设置成下列的位组合（不必考虑"X"的值）：

XX001100

把"X"项当作0，则可以将寄存器CCP1CON设置成十六进制数0CH。

载入到CCPR1L的值为"00010110"（即十六进制数16H）。

现将需要执行的步骤总结如下：

- 使用00000101（即05H）对预分频器值为4（即加载T2CON）的定时器2进行赋值；
- 将0AH载入到PR2；
- 将16H载入到CCPR1L；
- 将0载入到TRISC（将CCP1引脚配置为输出）；
- 将0CH载入到CCP1CON。

92

所产生的一个周期内的PWM波形如图2-39所示。

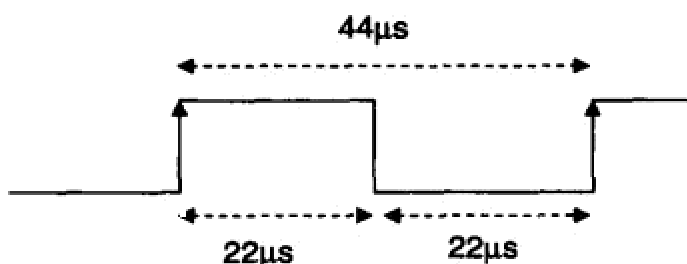


图2-39　生成的PWM波形

2.1.11　模数转换器（A/D）模块

模数转换器（A/D）是微控制器的又一个重要的外围部件。A/D转换器可以把一个模拟输入电压转换为一个能供微控制器或者任意其他的数字系统处理的数字值。在市场上有很多的模数转换器芯片，嵌入式系统的设计者应该理解这些芯片的特性，以高效地使用它们。

就输入和输出电压而言，A/D转换器可以分为单极性和双极性。单极性A/D转换器接受范围在0～+0 V的单极输入电压，而双极性A/D转换器接受范围在±V的双极输入电压。双极性A/D转换器经常应用于信号处理，自然地信号也是双极性的。单极性A/D转换器比较便宜，通常被应用在许多控制和仪器应用中。

93

图2-40介绍了将一个模拟信号读取和转换成数字信号的常用步骤。这个过程又被称作信号调理。从传感器上接收到的信号在传送到A/D转换器之前通常需要进行处理。该过程首先需要将模拟信号放大到合适值。接下来，使用经典的滤波器（如低通滤波器）将不需要的信号滤除。最后，通过一个抽样和保持设备，将信号传送给A/D转换器。这对于采样值不断变化的快速实时信号的处理尤其重要。抽样和保持设备将保证信号在实际转换过程中保持不变。许多应用都需要多个A/D转换器，因此通常在A/D转换器的输入端使用一个模拟多路复用器。多路复用器每次只选择一个信号，并把它传送给A/D转换器。A/D转换器通常有一个单独的模拟输入和一组数字的并行输出。其转换过程如下：

- 向A/D输入端施加处理过的信号；
- 启动转换；
- 等待，直到转换结束；
- 读取转换后的数字化数据。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

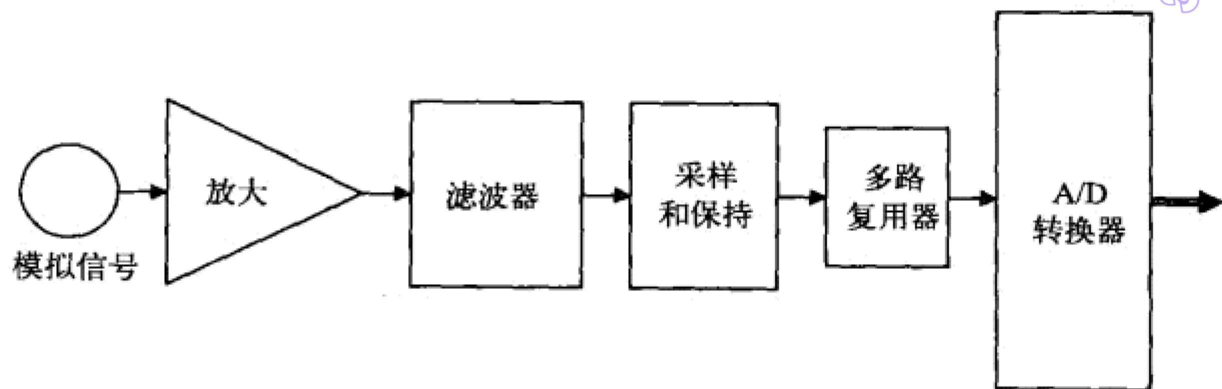


图2-40 信号调理和A/D转换过程

通过触发转换器来启动A/D转换。整个转换过程将花费几微秒，这取决于转换器的速度。在转换结束时，转换器将发出标志或者产生中断以表明转换已经结束。然后，转换得到的并行输出数据可被连接到A/D转换器上的数字设备读取。

大多数PIC18F系列微控制器都有一个10位的A/D转换器。如果选择的参考电压是+5 V，则单级电压值为：

$$\left(\frac{5\ \text{V}}{1\ 023}\right)=0.004\ 89\ \text{V}\ \text{即}\ 4.89\ \text{mV}$$

因此，如果输入电压是1.0 V，则转换器形成的十进制的数字输出是1.0/0.004 89=205。类似地，如果输入电压是3.0 V，那么转换器产生的数字输出是3.0/0.004 89=613。

PIC18F452微控制器使用的A/D转换器有8个通道，分别被命名为AN0～AN7，A/D转换器与PORTA和PORTE引脚共享它们。图2-41给出了A/D转换器的方框图。

94

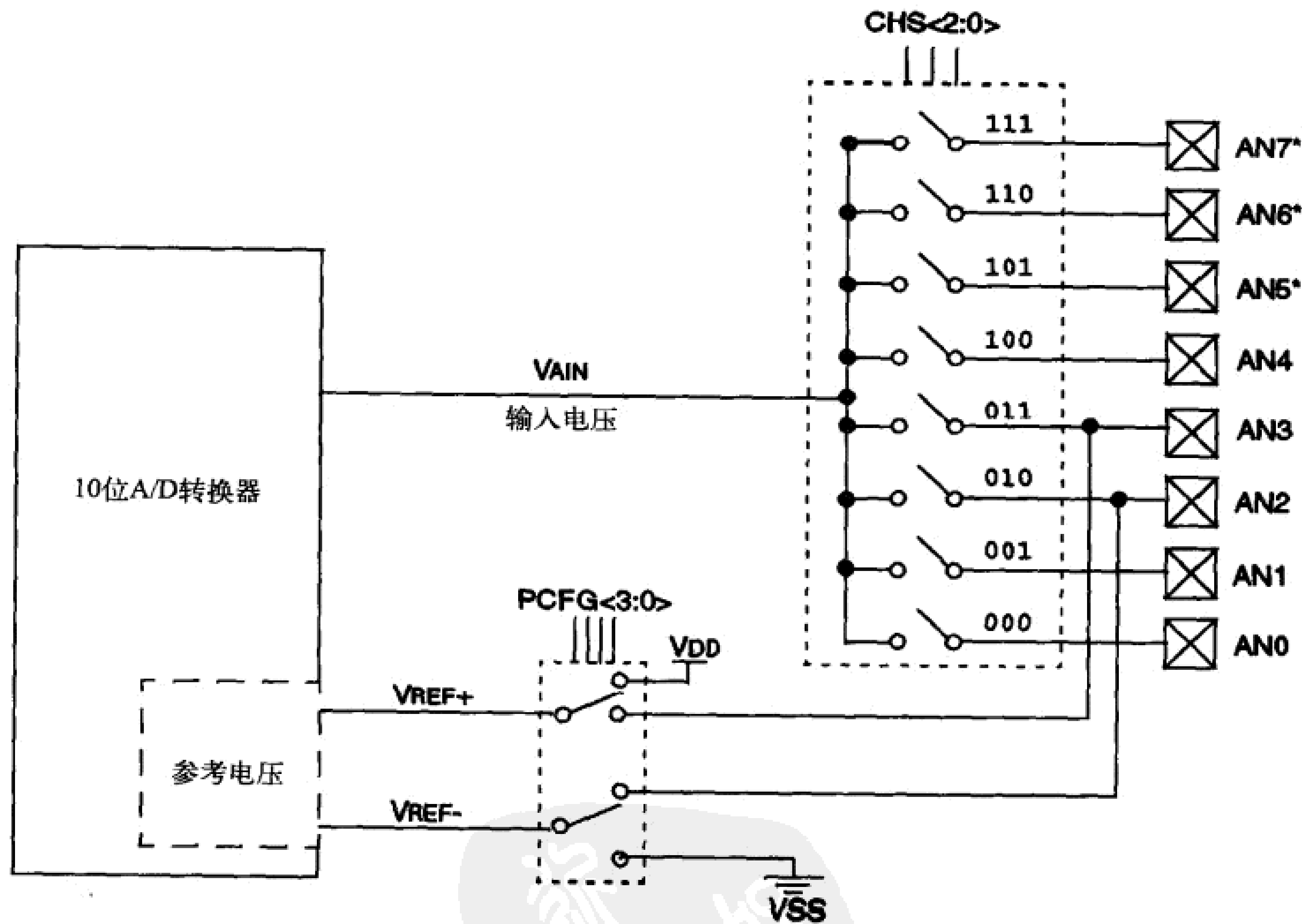
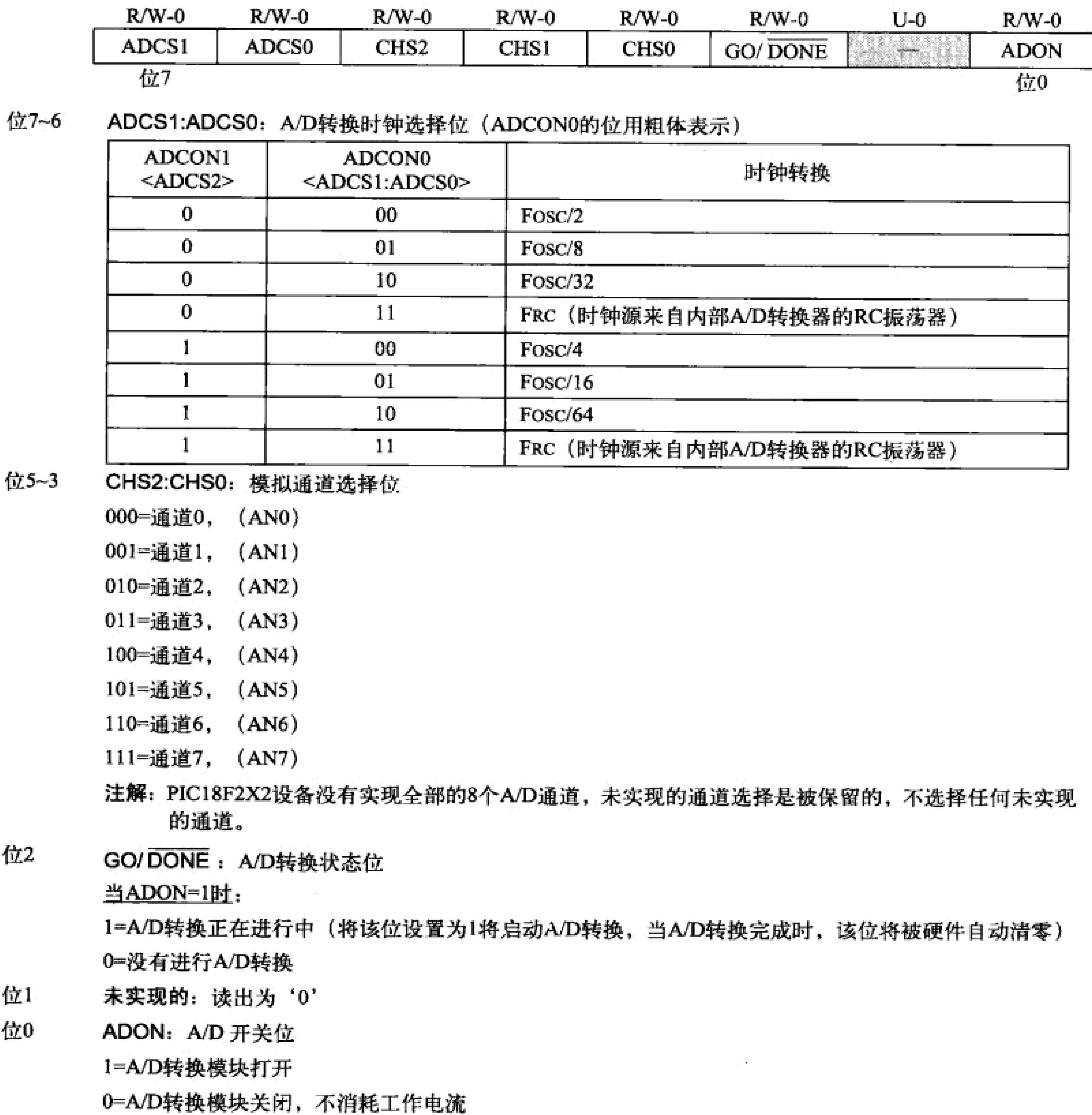


图2-41 PIC18F452 的A/D转换器的方框图

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

kyw藏书

A/D转换器有四个寄存器。寄存器ADRESH和ADRESL分别用来存储转换后的高字节和低字节结果。如图2-42所示，寄存器ADCON0用来控制A/D模块的运行，如和寄存器ADCON1一起选择时钟转换、选择输入通道、启动转换、打开和关闭A/D转换器。



R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/ $\overline{\text{DONE}}$	—	ADON
位7							位0

位7~6　ADCS1:ADCS0：A/D转换时钟选择位（ADCON0的位用粗体表示）

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	时钟转换
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	Frc（时钟源来自内部A/D转换器的RC振荡器）
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	Frc（时钟源来自内部A/D转换器的RC振荡器）

位5~3　CHS2:CHS0：模拟通道选择位

000=通道0，（AN0）
001=通道1，（AN1）
010=通道2，（AN2）
011=通道3，（AN3）
100=通道4，（AN4）
101=通道5，（AN5）
110=通道6，（AN6）
111=通道7，（AN7）

注解： PIC18F2X2设备没有实现全部的8个A/D通道，未实现的通道选择是被保留的，不选择任何未实现的通道。

位2　GO/ $\overline{\text{DONE}}$：A/D转换状态位

<u>当ADON=1时</u>：

1=A/D转换正在进行中（将该位设置为1将启动A/D转换，当A/D转换完成时，该位将被硬件自动清零）

0=没有进行A/D转换

位1　未实现的：读出为‘0’

位0　ADON：A/D 开关位

1=A/D转换模块打开

0=A/D转换模块关闭，不消耗工作电流

图2-42 ADCON0寄存器

寄存器ADCON1（如图2-43所示）用来选择转换格式、将A/D转换通道配置为模拟输入、选择参考电压、同寄存器ADCON0一起选择时钟转换。

95 ~ 96

可通过置位寄存器ADCON0的GO/DONE位启动A/D转换。当转换完成时，转换所得数据的2位被写入寄存器ADRESH，而其余的8位被写入寄存器ADRESL。同时，GO/DONE位被清零，以表明转换结束。如果有需要，还可以使能中断，当转换完成时将产生一个软件中断。

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
位7							位0

位7　ADFM：A/D转换结果格式选择位

1=右对齐。ADRESH的6个最高有效位读出为“0”

0=左对齐。ADRESL的6个最低有效位读出为“0”

位6　ADCS2：A/D转换时钟选择位（ADCON1的位用粗体表示）

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	时钟转换
0	00	$F_{osc/2}$
0	01	$F_{osc/8}$
0	10	$F_{osc/32}$
0	11	F_{RC}（时钟源来自内部A/D转换器的RC振荡器）
1	00	$F_{osc/4}$
1	01	$F_{osc/16}$
1	10	$F_{osc/64}$
1	11	F_{RC}（时钟源来自内部A/D转换器的RC振荡器）

位5~4　未实现的：读出为‘0’

位3~0　PCFG3:PCFG0：A/D端口配置控制位

PCFG <3：0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

A=模拟输入　　D=数字I/O

图2-43　ADCON1寄存器

97

一次A/D转换的完整步骤如下：

- 使用寄存器ADCON1将期望的通道配置为模拟，并设置参考电压；
- 将TRISA或TRISE位置1，使得期望的通道为输入端口；
- 使用寄存器ADCON0来选择期望的模拟输入通道；
- 使用寄存器ADCON0和ADCON1来选择转换时钟；
- 使用寄存器ADCON0来打开A/D模块；

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

tyw藏书

- 配置A/D中断（如果有必要）；
- 将GO/DONE位置位以启动转换；
- 等待，直到GO/DONE位被清零或者转换完成时产生中断；
- 从ADRESH和ADRESL中读取转换后的数据；
- 根据需要，重复上述的步骤。

为了得到正确的A/D转换结果，必须选择合适的A/D转换时钟来保证位转换时间最短（为 98 1.6 μs）。表2-11给出了不同的微控制器工作频率所需的A/D时钟源。例如，如果微控制器工作在10 MHz的时钟上，那么A/D时钟源应该为$F_{OSC/16}$或者更高（如$F_{OSC/32}$）。

表2-11 A/D转换时钟选择

A/D时钟源		最高微控制器频率
转换时间	ADCS2：ADCS0	
2 T_{OSC}	000	1.25 MHz
4 T_{OSC}	100	2.5 MHz
8 T_{OSC}	001	5.0 MHz
16 T_{OSC}	101	10.0 MHz
32 T_{OSC}	010	20.0 MHz
64 T_{OSC}	110	40.0 MHz
RC	011	—

寄存器ADCON1的位ADFM用来控制转换结果的格式。当ADFM被清零时，10位结果是左对齐的（如图2-44所示）且ADRESL的低6位被清零。当ADFM被置1时，10位结果将是右对齐的且ADRESH的高6位被清零。这是最常用的模式，其中ADRESL包含低8位，ADRESH的位0和位1包含10位结果的高2位。

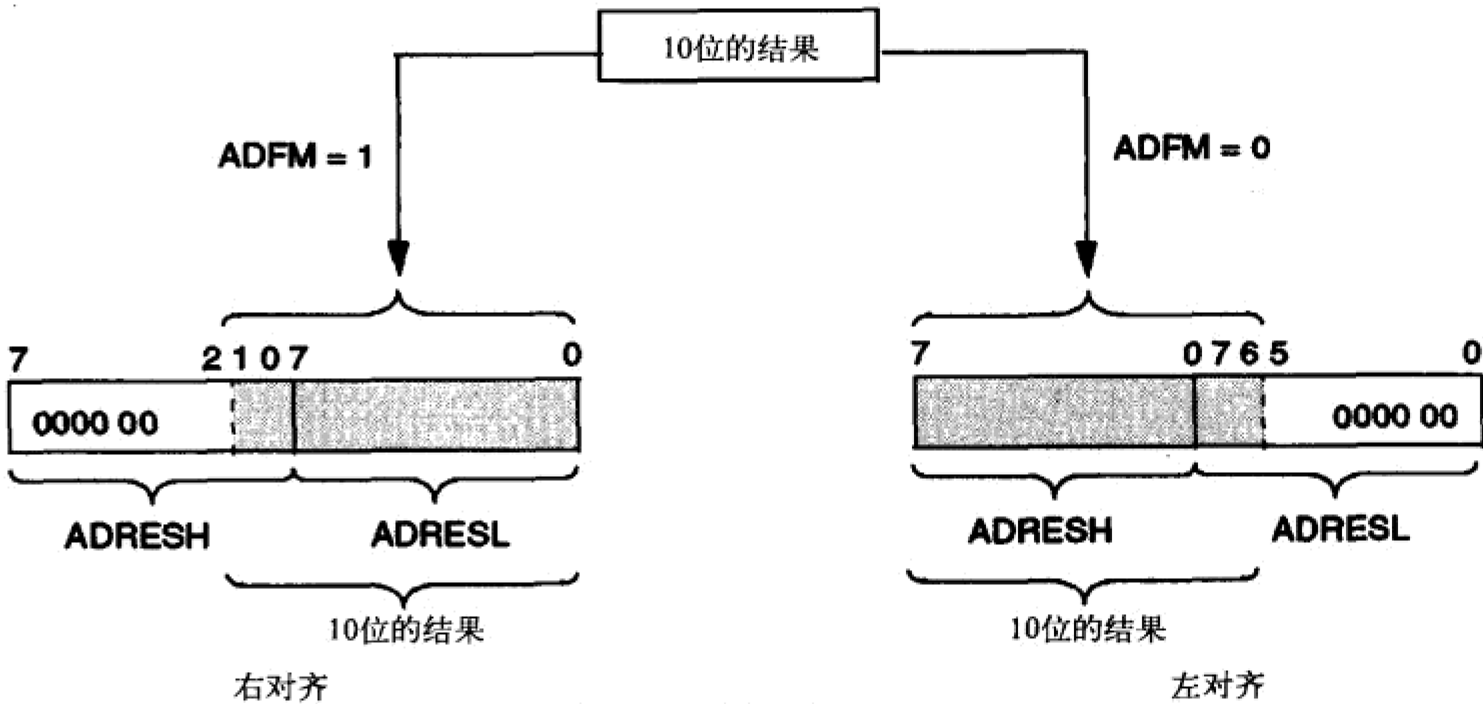


图2-44 A/D转换结果的格式化

模拟输入模型和采集时间

理解A/D模拟输入模型对于如何将A/D转换器接到外部设备是很有必要的。图2-45所示的是A/D转换器的模拟输入模型。图2-45左边所示的是模拟输入电压V_{AIN}和内电阻R_S。建议内电阻不能大于2.5 kΩ。模拟信号应施加到标有ANX的引脚上。还有一个小电容（5 pF），对地的漏电流 99 近似为500 nA。RIC为连接电阻，它的值小于1 kΩ。在采样过程中，采样开关SS的电阻R_{SS}取决

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

于电压，其函数曲线如图2-45底部的小图所示。在电源电压为5 V时，*RSS*的值大约为7 kΩ。

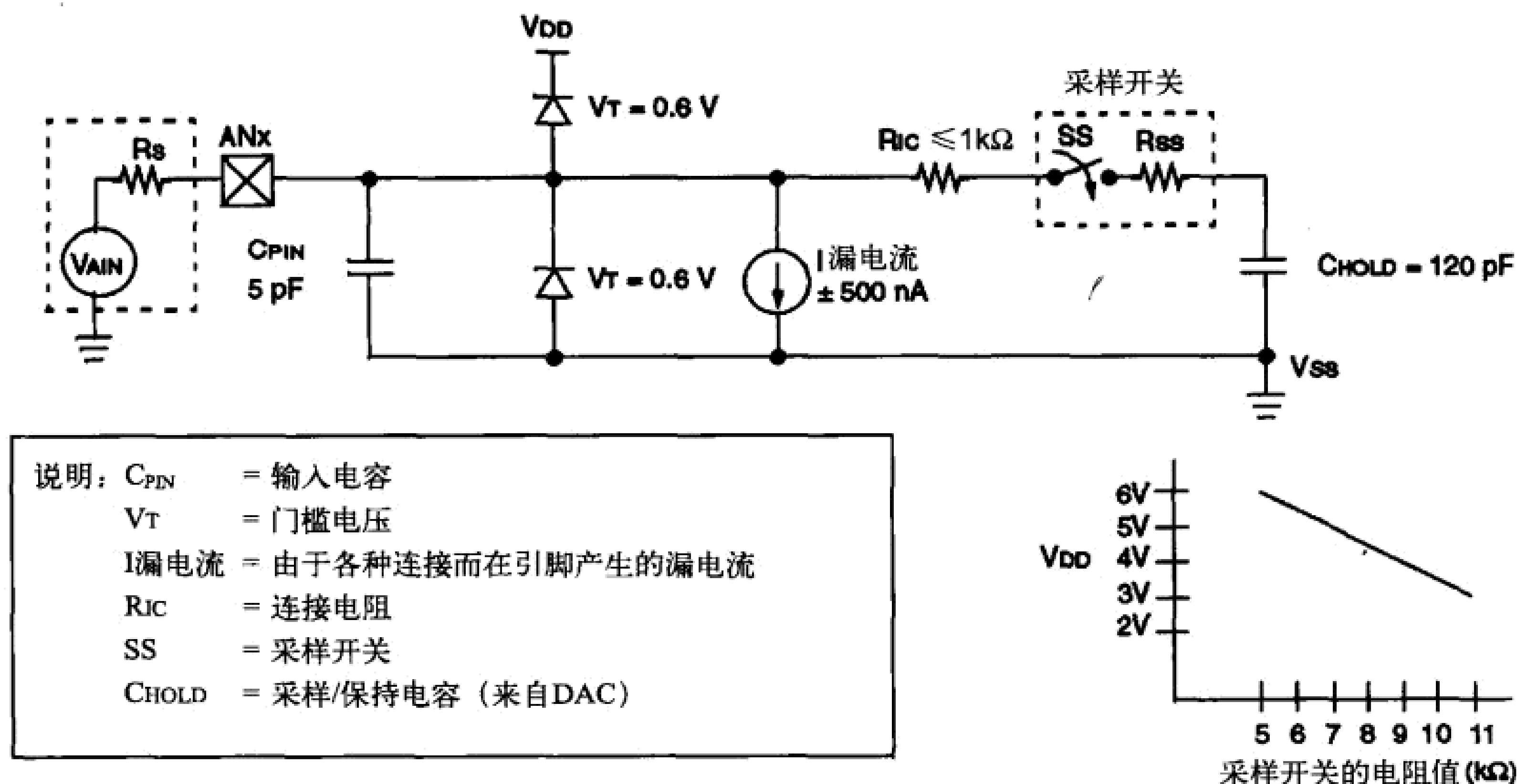


图2-45 A/D转换器的模拟输入模型

A/D转换器是基于开关电容原理的，图2-45所示的电容C_{HOLD}在开始转换之前必须充满电。这里使用一个120 pF的电容，一旦转换开始，它就会从引脚上断开。

采集时间的计算可以使用Microchip公司提供的式（2.7）：

$$T_{ACQ}=\text{放大器稳定时间}+\text{保持电容充电时间}+\text{温度系数} \quad (2.7)$$

放大器稳定时间被指定为固定的2 μs。温度系数只有在温度超过25 ℃时才有效，其计算公式如下：

$$\text{温度系数}=(\text{温度}-25\ ℃)(0.05\ \mu s/℃) \quad (2.8)$$

式（2.8）表明，温度的影响是很小的，在25 ℃以上每超过10 ℃将会产生0.5 μs的延迟。因此，假设工作环境在25 ℃～35 ℃之间，由于温度升高而产生的延迟时间最大为0.5 μs。这个在大多数实际应用中是可以忽略的。 100

由Microchip公司指定的保持电容充电时间为：

$$\text{保持电容充电时间}=-(120\text{pF})(1\ k\Omega+R_{SS}+R_S)Ln(1/2\ 048) \quad (2.9)$$

假设R_{SS}=7 kΩ、R_S=2.5 kΩ，式（2.9）计算出的保持电容充电时间为9.6 μs。

因此，可得到的采集时间如下：

$$T_{ACQ}=2+9.6+0.5=12.1\ \mu s$$

一个满10位的转换需要12个A/D周期，每个A/D周期最少需要1.6 μs。因此，最快的转换时间为19.2 μs。再加上最好的采集时间，则最终得到的转换时间总共为19.2＋12.1=31.3 μs。

当一次转换完成时，转换器在开始一次新的转换之前需要等待两个转换周期。这个时间相当于2×1.6=3.2 μs。再将这个时间加到31.3 μs的最佳可能转换时间上，则可得到完成转换的时间为34.5 μs。假设A/D转换器成功使用，并且忽略软件的时间开销，这意味着最大的采样频率大约是29 kHz。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

2.1.12　中断

中断是指一个事件，要求CPU停止正常的程序执行，转而执行与引发中断的事件相关的程序代码。中断可以由内部产生（由芯片内的一些事件引起）或者外部产生（由一些外部事件引起）。定时器溢出或者A/D转换结束产生的中断是内部中断，而I/O引脚状态改变产生的中断就是一个外部中断的例子。

中断在许多应用中都是很有用的，如下所示。

- **时间紧急应用**。在需要CPU立即干预的应用中可以使用中断。例如，在发生断电或者起火之类的紧急事件，CPU必须以一种有序的方式立即关断系统。在这种应用中，不管CPU正在执行什么任务，外部中断都将强制CPU停止工作，并立即采取行动。
- **执行例行任务**。许多应用要求CPU在精确的时间执行例行的工作，例如每毫秒精确地检测外围设备的状态。根据时间调度，定时器中断可用来将CPU从正常的程序执行转向在期望的精确时间执行其他的任务。
- **多任务应用中的任务切换**。在多任务应用中，每个任务都可能有一个限定时间去执行它的程序代码。中断机制可用来终止那些消耗时间多于分配时间的任务。
- **快速地维护外围设备**。有些应用需要知道任务的完成时间，如A/D转换。这可以通过不断地检查A/D转换器的完成标志来实现。更完美的方法是使能A/D转换完成中断，只要转换完成，就可以强制CPU读取转换所得的数据。

[101]

PIC18F452微控制器既有核心中断源，也有外围中断源。核心中断源为：

- 在INT0、INT1和INT2引脚上的外部边沿触发中断；
- PORTB引脚变化中断（RB4～RB7引脚中任意一个引脚状态发生变化）；
- 定时器0溢出中断。

外围中断源为：

- 并行从端口读/写中断；
- A/D转换完成中断；
- USART接收中断；
- USART发送中断；
- 同步串行端口中断；
- CCP1中断；
- TMR1溢出中断；
- TMR2溢出中断；
- 比较器中断；
- EEPROM/FLASH写中断；
- 总线冲突中断；
- 低压检测中断；
- 定时器3溢出中断；
- CCP2中断。

[102]

在PIC18F系列中，中断被分为两组：高优先级和低优先级。需要更多注意的应用可以放在高优先级的组中。一个高优先级中断能终止一个正在执行的低优先级中断，从而获得对CPU的使用权，而低优先级中断不能终止高先级中断。如果应用对中断不需要设置优先级，那么用户可以选择禁止优先级的方案，从而使所有的中断都处于同一个优先级。在程序存储器中，高优

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

先级中断向量指向地址00008H，而低优先级中断向量指向地址000018H。通常，用户程序代码（ISR，中断服务子例程）应该放置在中断向量地址处，以服务中断设备。

在PIC18F452微控制器中，有10个用来控制中断操作的寄存器。它们是：

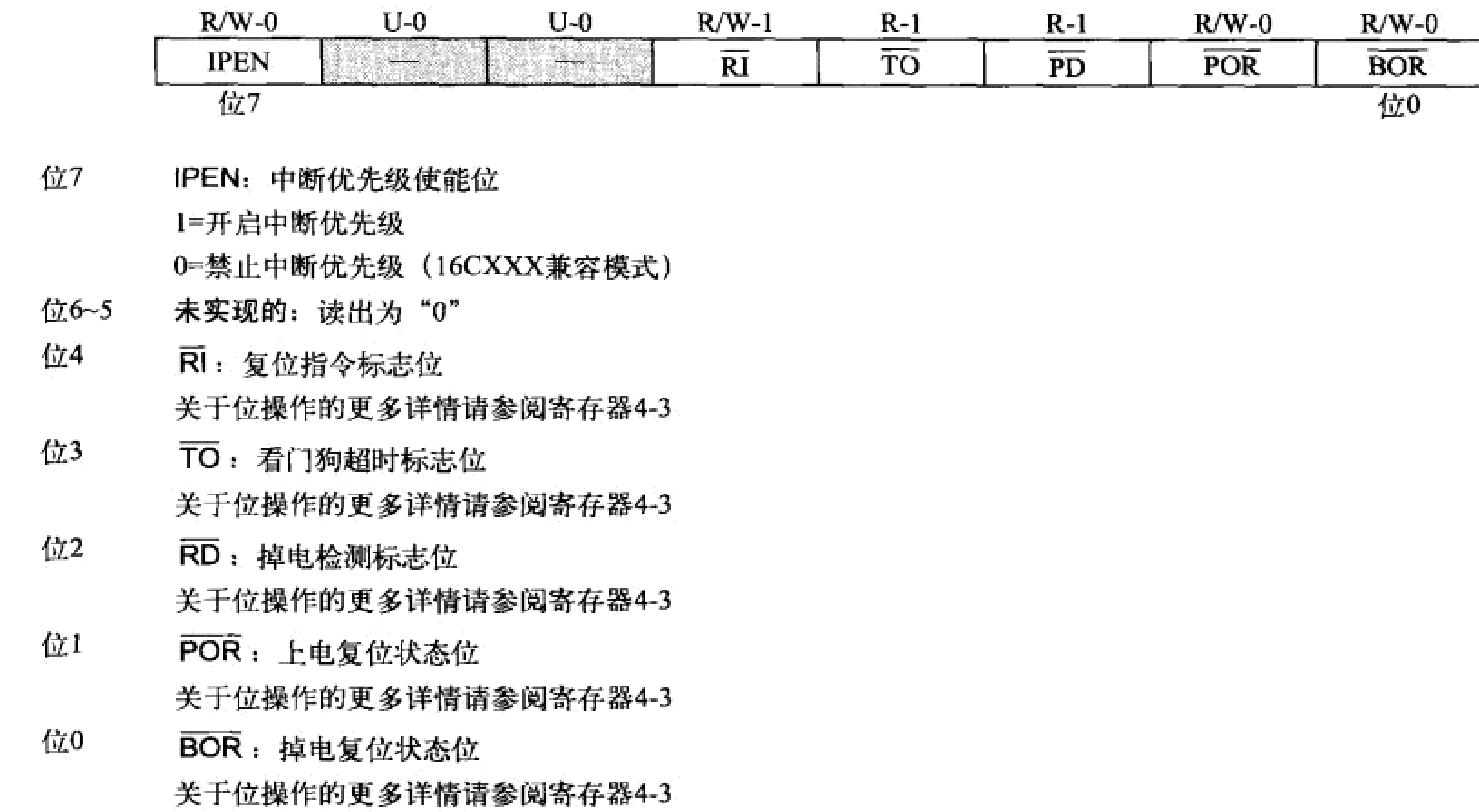
- RCON寄存器；
- INTCON寄存器；
- INTCON2寄存器；
- INTCON3寄存器；
- PIR1、PIR2寄存器；
- PIE1、PIE2寄存器；
- IPR1、IPR2寄存器。

每个中断源（除了INT0）都有3个位用来控制中断操作。这3个位如下所示。

- 一个标志位，用来表明中断是否发生。该位的名字以IF结尾。
- 一个中断使能位，用来决定中断源是否有效。该位的名字以IE结尾。 103
- 一个优先级位，用来选择高优先级或者低优先级。该位的名字以IP结尾。

1. RCON寄存器

RCON寄存器的最高位叫作IPEN，用来启用中断优先级方案。当IPEN=0时，中断优先级被禁止，微控制器的中断结构和PIC16系列很相似。当IPEN=1是，中断优先级被启用。图2-46所示的是寄存器RCON的位描述。



R/W-0	U-0	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
位7							位0

位7 IPEN：中断优先级使能位
1=开启中断优先级
0=禁止中断优先级（16CXXX兼容模式）

位6~5 未实现的：读出为“0”

位4 $\overline{RI}$：复位指令标志位
关于位操作的更多详情请参阅寄存器4-3

位3 $\overline{TO}$：看门狗超时标志位
关于位操作的更多详情请参阅寄存器4-3

位2 $\overline{RD}$：掉电检测标志位
关于位操作的更多详情请参阅寄存器4-3

位1 $\overline{POR}$：上电复位状态位
关于位操作的更多详情请参阅寄存器4-3

位0 $\overline{BOR}$：掉电复位状态位
关于位操作的更多详情请参阅寄存器4-3

图2-46 RCON寄存器的位描述

2. 启用/禁止中断——无优先级结构

当IPEN位被清零时，优先级结构是被禁止的。所有中断都跳转到程序存储器的地址00008H。在这种模式下，寄存器INTCON的位PEIE用来启用/禁止所有外围中断源。相似地，寄存器INTCON的位GIE用来启用/禁止所有的中断源。图2-47所示的是寄存器INTCON的位描述。 104

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
位7							位0

位7　GIE/GIEH：全局中断使能位

当IPEN=0时，

1=启用所有未屏蔽中断

0=关闭所有中断

当IPEN=1时，

1=启用所有高优先级中断

0=关闭所有中断

位6　PEIE/GIEL：外围中断使能位

当IPEN=0时，

1=启用所有未屏蔽的外围中断

0=关闭所有外围中断

当IPEN=1时，

1=启用所有的低优先级外围中断

0=关闭所有的低优先级外围中断

位5　TMR0IE：TMR0溢出中断使能位

1=启用TMR0溢出中断

0=禁止TMR0溢出中断

位4　INT0IE：INT0外部中断使能位

1=启用INT0外部中断

0=禁止INT0外部中断

位3　RBIE：RB端口变化中断使能位

1=启用RB端口变化中断

0=禁止RB端口变化中断

位2　TMR0IF：TMR0溢出中断标志位

1=TMR0寄存器已经溢出（必须用软件清零）

0=TMR0寄存器没有溢出

位1　INT0IF：INT0外部中断标志位

1=有INT0外部中断发生（必须用软件清零）

0=没有INT0外部中断发生

位0　RBIF：RB端口变化中断标志位

1=RB7:RB4引脚中至少有一个状态发生改变（必须用软件清零）

0=RB7:RB4引脚状态没有变化

注解：一个不匹配的条件将用来将该位置1。读PORTB可以结束不匹配的条件，并且允许该位被清零。

105

图2-47　INTCON寄存器的位描述

若要CPU接受一个中断，必须满足下面的条件。

- 必须启用中断源的中断使能位。例如，如果中断源是外部中断引脚INT0，那么必须将寄存器INTCON的位INT0IE置为1。
- 必须将中断源的中断标志清零。例如，如果中断源是外部中断引脚INT0，那么必须将寄存器INTCON的位INT0IF必须清零。
- 如果中断源是一个外围设备，则必须将寄存器INTCON的外部中断启用/禁止位PEIE置为1。
- 必须将寄存器INTCON的全局中断启用/禁止位GIE置为1。

当使用外部中断源时，通常必须定义中断源的发生是在由低电平到高电平的上升沿还是由高电平到低电平的下降沿。例如，如果使用INT0中断，可以通过置位/清零寄存器INTCON2的

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

位INTEDG0来选择。

发生中断时，CPU会停止执行正常流程，将返回地址压入栈，并跳转到程序存储器的地址00008H，这是用户中断服务子程序驻留的首地址。一旦CPU开始执行中断服务子程序，则全局中断使能位（GIE）将被清零，以禁止其他的中断请求。当多个中断源被启用时，中断源可以通过轮询中断标志位来决定。在再次使能中断之前，中断标志位必须使用软件清零以避免递归中断。当CPU已经从中断服务子程序返回时，全局中断位GIE就由软件自动地置1。

3. 启用/禁止中断——优先级结构

将IPEN位设置为1时，优先级特性被启用，并且中断被分为两组：低优先级和高优先级。低优先级中断指向程序存储器地址00008H，而高优先级中断指向程序存储器地址000018H。将优先级位置1，可以将中断源编程为高优先级中断，而将该位清零，可以将中断源编程为低优先级中断。 106

将寄存器INTCON的GIEH位置1，用来启用所有的高优先级中断，即优先级位被置1。相似地，将寄存器INTCON的GIEL位置1，用来启用所有的低优先级中断（优先级位被清零）。

若要使CPU接受一个高优先级的中断，必须满足下列条件。

- 必须启用中断源的中断使能位。例如，如果中断源是外部中断引脚INT1，那么必须将寄存器INTCON3的位INT1IE设置为1。
- 将中断源的中断标志位清零。例如，如果中断源是外部中断引脚INT1，那么寄存器INTCON3的位INT1IF必须被清零。
- 必须将优先级位置为1。例如，如果中断源是外部中断引脚INT1，那么必须将寄存器INTCON3的位INT1IP置为1。
- 必须将寄存器INTCON的全局中断启用/禁止位GIEH置为1。

若要CPU接受一个低优先级的中断，必须满足下列条件。

- 必须启用中断源的中断使能位。例如，如果中断源是外部中断引脚INT1，那么必须将寄存器INTCON3的位INT1IE置为1。
- 将中断源的中断标志位清零。例如，如果中断源是外部中断引脚INT1，那么必须将寄存器INTCON3的位INT1IF清零。
- 必须将优先级位清零。例如，如果中断源是外部中断INT1，那么必须将寄存器INTCON3的位INT1IP清零。
- 低优先级中断必须通过将寄存器INTCON的GIEL位置1来启用。
- 必须将寄存器INTCON的全局中断启用/禁止位GIEH置为1。

表2-12罗列了PIC18F452微控制器的每个中断源的中断位名字和寄存器名字。 107

表2-12 PIC18F452的中断位和寄存器

中断源	标志位	使能位	优先级位
INT0外部	INT0IF	INT0IE	—
INT1外部	INT1IF	INT1IE	INT1IP
INT2外部	INT2IF	INT2IE	INT2IP
RB端口变化	RBIF	RBIE	RBIP
TMR0溢出	TMR0IF	TMR0IE	TMR0IP
TMR1溢出	TMR1IF	TMR1IE	TMR1IP
TMR2匹配PR2	TMR2IF	TMR2IE	TMR2IP
TMR3溢出	TMR3IF	TMR3IE	TMR3IP

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源

tyw藏书

（续）

中断源	标志位	使能位	优先级位
A/D转换完成	ADIF	ADIE	ADIP
CCP1	CCP1IF	CCP1IE	CCP1IP
CCP2	CCP2IF	CCP2IE	CCP2IP
USART RCV	RCIF	RCIE	RCIP
USART TX	TXIF	TXIE	TXIP
并行从端口	PSPIF	PSPIE	PSPIP
同步串行端口	SSPIF	SSPIE	SSPIP
低电压检测	LVDIF	LVDIE	LVLIP
总线冲突	BCLIF	BCLIE	BCLIP
EEPROM/FLASH写	EEIF	EEIE	EEIP

图2-48～图2-55给出了中断寄存器INTCON2、INTCON3、PIR1、PIR2、PIE1、PIE2、IPR1和IPR2的位定义。

这部分将给出许多的例子来说明如何将CPU编程用于中断。

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
位7							位0

位7　$\overline{\text{RBPU}}$：PORTB上拉使能位

1=所有的PORTB上拉被禁止

0=PORTB上拉通过独立的端口锁存值被启用

位6　INTEDG0：外部中断0边沿选择位

1=在上升沿中断

0=在下降沿中断

位5　INTEDG1：外部中断1边沿选择位

1=在上升沿中断

0=在下降沿中断

位4　INTEDG2：外部中断2边沿选择位

1=在上升沿中断

0=在下降沿中断

位3　未实现：读出为‘0’

位2　TMR0IP：TMR0溢出中断优先级位

1=高优先级

0=低优先级

位1　未实现：读出为‘0’

位0　RBIP：RB端口变化中断优先级位

1=高优先级

0=低优先级

图2-48　寄存器INTCON2的位定义

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

R/W-1	R/W-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF
位7							位0

位7　INT2IP：INT2外部中断优先级位
1=高优先级
0=低优先级

位6　INT1IP：INT1外部中断优先位
1=高优先级
0=低优先级

位5　未实现：读出为“0”

位4　INT2IE：INT2外部中断使能位
1=启用INT2外部中断
0=禁止INT2外部中断

位3　INT1IE：INT1外部中断使能位
1=启用INT1外部中断
0=禁止INT1外部中断

位2　未实现：读出为“0”

位1　INT2IF：INT2外部中断标志位
1=INT2外部中断发生（必须用软件清零）
0=INT2外部中断没有发生

位0　INT1IF：INT1外部中断标志位
1=INT1外部中断发生（必须用软件清零）
0=INT1外部中断没有发生

图2-49　寄存器INTCON3的位定义

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
位7							位0

位7　PSPIF：并行从端口读/写中断标志位
1=一个读或者写操作已经发生（必须用软件清零）
0=没有读或者写操作发生

位6　ADIF：A/D转换器中断标志位
1=一个A/D转换完成（必须用软件清零）
0=A/D转换没有完成

位5　RCIF：USART接收中断标志位
1=USART接收缓冲器RCREG已满（当读RCREG时，缓冲器被清零）
0=USART接收缓冲器是空的

位4　TXIF：USART传输中断标志位
1=USART发送缓冲器TXREG已空（当写入TXREG时，缓冲器被清零）
0=USART发送缓冲器是满的

位3　SSPIF：主同步串行端口中断标志位
1=传输/接收完成（必须用软件清零）
0=等待传输/接收

位2　CCP1IF：CCP1中断标志位
<u>捕捉模式</u>：
1=一个TMR1寄存器捕捉发生（必须用软件清零）
0=没有TMR1寄存器捕捉发生
<u>比较模式</u>：
1=一个TMR1寄存器比较匹配发生（必须用软件清零）
0=没有TMR1寄存器比较匹配发生
<u>PWM模式</u>：
在该模式中没有被使用

位1　TMR2IF：TMR2与PR2匹配中断标志位
1=TMR2与PR2匹配发生（必须用软件清零）
0=没有TMR2与PR2匹配发生

位0　TMR1IF：TMR1溢出中断标志位
1=TMR1寄存器溢出（必须用软件清零）
0=TMR1寄存器没有溢出

图2-50　寄存器P1R1的位定义

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF
位7							位0

位7~5 未实现：读出为“0”

位4 EEIF：数据EEPROM/FLASH写操作中断标志位
1=写操作完成（必须用软件清零）
0=写操作没有完成，或者还没有开始

位3 BCLIF：总线冲突标志位
1=一个总线冲突发生（必须用软件清零）
0=没有总线冲突发生

位2 LVDIF：低电压检测中断标志位
1=一个低压条件发生（必须用软件清零）
0=设备电压大于低压检测跳变点

位1 TMR3IF：TMR3溢出中断标志位
1=TMR3寄存器溢出（必须用软件清零）
0=TMR3寄存器没有溢出

位0 CCP2IF：CCPx中断标志位
捕捉模式：
1=一个TMR1寄存器捕捉发生（必须用软件清零）
0=没有TMR1寄存器捕捉发生
比较模式：
1=一个TMR1寄存器比较匹配发生（必须用软件清零）
0=没有TMR1寄存器比较匹配发生
PWM模式：
在该模式中未被使用

图2-51 寄存器PIR2的位定义

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
位7							位0

位7 PSPIE：并行从端口读/写中断使能位
1=启用PSP读/写中断
0=禁止PSP读/写中断

位6 ADIE：A/D转换器中断使能位
1=启用A/D中断
0=禁止A/D中断

位5 RCIE：USART接收中断使能位
1=启用USART接收中断
0=禁止USART接收中断

位4 TXIE：USART传输中断使能位
1=启用USART传输中断
0=禁止USART传输中断

位3 SSPIE：主同步串行端口中断使能位
1=启用MSSP中断
0=禁止MSSP中断

位2 CCP1IE：CCP1中断使能位
1=启用CCP1中断
0=禁止CCP1中断

位1 TMR2IE：TMR2与PR2匹配中断使能位
1=启用TMR2与PR2匹配中断
0=禁止TMR2与PR2匹配中断

位0 TMR1IE：TMR1溢出中断使能位
1=启用TMR1溢出中断
0=禁止TMR1溢出中断

图2-52 寄存器PIE1的位定义

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE
位7							位0

位7～5 未实现：读出为“0”
位4 EEIE：数据EEPROM/FLASH写操作中断使能位
1=启用
0=禁止
位3 BCLIE：总线冲突使能位
1=启用
0=禁止
位2 LVDIE：低压检测中断使能位
1=启用
0=禁止
位1 TMR3IE：TMR3溢出中断使能位
1=启用TMR3溢出中断
0=禁止TMR3溢出中断
位0 CCP2IE：CCP2中断使能位
1=启用CCP2中断
0=禁止CCP2中断

图2-53 寄存器PIE2的位定义

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
位7							位0

位7 PSPIP：并行从端口读/写中断优先级位
1=高优先级
0=低优先级
位6 ADIP：A/D转换器中断优先级位
1=高优先级
0=低优先级
位5 RCIP：USART接收中断优先级位
1=高优先级
0=低优先级
位4 TXIP：USART传输中断优先级位
1=高优先级
0=低优先级
位3 SSPIP：主同步串行端口中断优先级位
1=高优先级
0=低优先级
位2 CCP1IP：CCP1中断优先级位
1=高优先级
0=低优先级
位1 TMR2IP：TMR2与PR2匹配中断优先级位
1=高优先级
0=低优先级
位0 TMR1IP：TMR1溢出中断优先级位
1=高优先级
0=低优先级

图2-54 寄存器IPR1的位定义

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

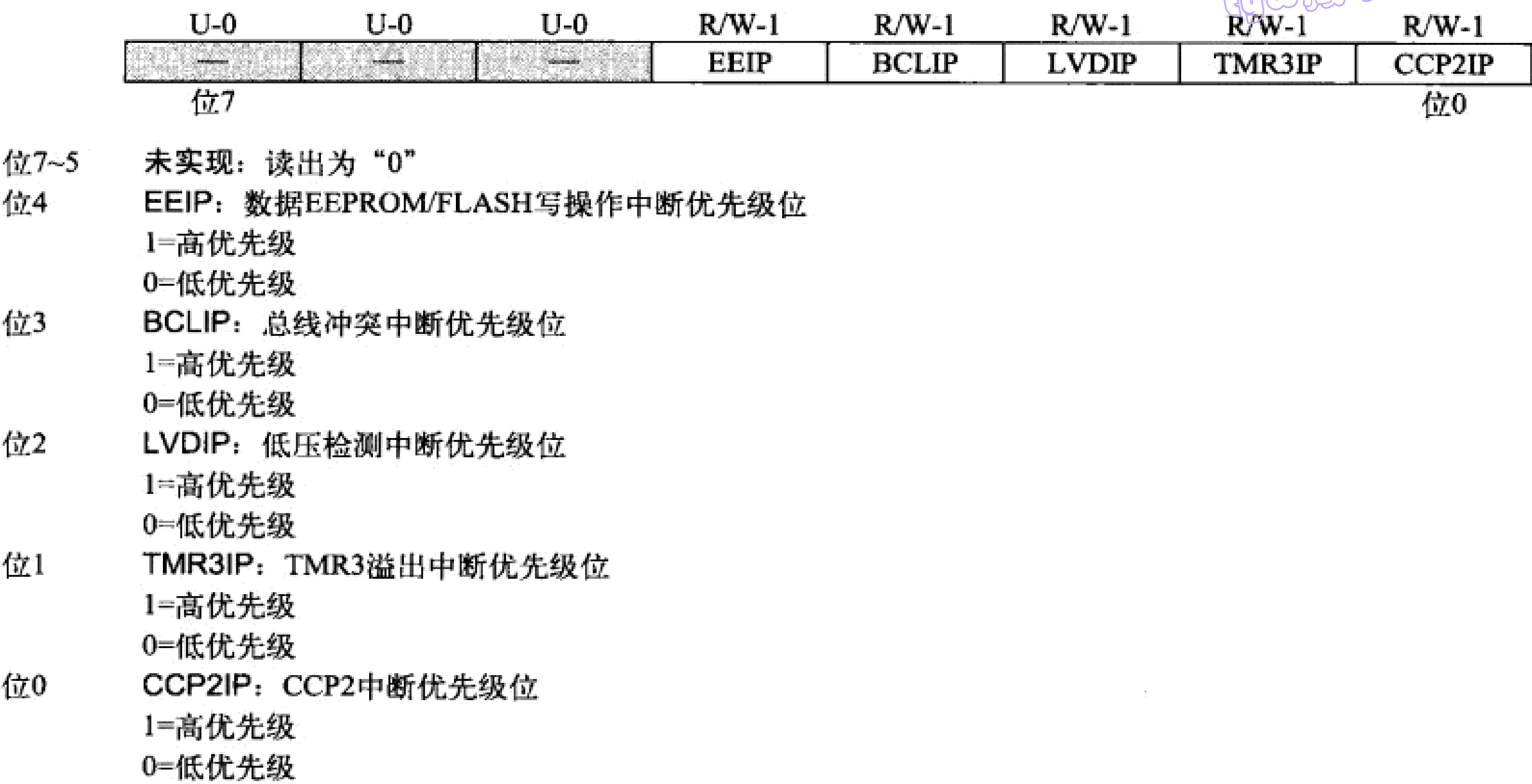


图2-55　寄存器IPR2的位定义

108~109

例2.2　将INT1设置为一个低优先级的下降沿触发的中断输入。

解　在低优先级模式下，要使INT1下降沿触发的中断能够被CPU所接受，必须设置下面的各位。

- 启用优先级结构。设置IPEN=1。
- 将INT1配置为输入引脚。设置TRISB=1。
- 将INT1中断设置为下降沿触发。设置INTEDG1=0。
- 启用INT1中断。设置INT1IE=1。
- 启用低优先级。设置INT1IP=0。
- 将INT1 标志清零。设置INT1IF=0。
- 启用低优先级中断。设置GIEL=1。
- 启用所有中断。设置GIEL=1。

110~111

发生中断时，CPU就跳转到程序存储器的地址00008H，执行在中断服务例程中的用户程序。

例2.3　将INT1设置为一个高优先级的上升沿触发的中断。

解　在高优先级模式下，要使INT1上升沿触发的中断能够被CPU接受，则必须设置下面的各位。

- 启用优先级结构。设置IPEN=1。
- 将INT1配置为输入引脚。设置TRISB=1。
- 将INT1中断设置为上升沿触发。设置INTEDG1=1。
- 启用INT1中断。设置INT1IE=1。
- 启用高优先级。设置INT1IP=1。
- 将INT1标志清零。设置INT1IF=1。
- 启用所有中断。设置GIEH=1。

112

113~114

发生中断时，CPU就会跳转到程序存储器的地址000018H，来执行在中断服务例程中的用户程序。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

2.2 小结

本章介绍了PIC18F系列微控制器的结构。PIC18F452被看作本系列微控制器的代表。同系列的其他成员，例如PIC18F242具有更少的引脚、更少的功能。另外一些微控制器，例如PIC18F6680具有更多的引脚和更多的功能。

本章还介绍了PIC18F系列的重要模块和外围电路，如数据存储器、程序存储器、时钟电路、复位电路、看门狗定时器、通用目的定时器、捕捉和比较模块、PWM模块、A/D转换器和中断结构等。

2.3 练习题

1. 请描述PIC18F452微控制器的数据存储器结构。什么是存储区？有多少个存储区？
2. 请解释通用目的寄存器（GPR）和特殊功能寄存器（SFR）之间的区别。 115
3. 请说明PIC18F微控制器的几种复位方法。绘制电路图来说明如何使用外部按钮开关来复位微控制器。
4. 请描述用来为PIC18F452微控制器提供时钟的几种时钟源。绘制电路图来说明如何将一个10MHz的晶体连接到微控制器。
5. 请绘制电路图来说明如何将谐振器连接到PIC18F微控制器。
6. 在时间不是很重要的应用中，必须使用外部电阻和电容来为PIC18F452微控制器提供时钟。绘制电路图来说明如何实现时钟频率为5MHz的电路，并计算出这两个元件的值。
7. 请说明如何使用外部时钟向PIC18F微控制器提供时钟脉冲。
8. PORTA的寄存器有哪些？请绘制端口方框图来说明端口的操作。
9. 看门狗定时器必须设置为每0.5秒提供一个自动复位。请阐述如何实现之，包括合适的寄存器位的选择和操作。
10. PWM脉冲必须从PIC18F452微控制器的引脚CCP1产生。要求脉冲周期为100μs，并且要求占空比周期为50%。假设微控制器的晶振频率为4MHz，计算载入各种寄存器的初值。
11. 同样，PWM脉冲从PIC18F452微控制器的引脚CCP1产生。如果要求脉冲频率为40KHz，并且要求占空比周期为50%。假设微控制器的晶振频率为4MHz，计算载入各种寄存器的初值。
12. 将一个LM35DZ型的模拟温度传感器连接到PIC18F452微控制器的模拟端口AN0。传感器输出和温度成比例的模拟电压（即V0=10mV/℃）。请给出读取温度数据所需的步骤。
13. 请说明优先级中断和非优先级中断之间的区别。 116
14. 请给出将INT2设置为一个具有低优先级的下降沿触发的中断输入所需的步骤。中断向量地址是多少？
15. 请给出将INT1和INT2设置为具有低优先级的下降沿触发的中断输入所需的步骤。
16. 将INT1设置为下降沿触发的中断输入，将INT2设置为上升沿触发的中断输入，且都具有高优先级，请给出步骤。并说明当一个中断发生时是如何确定中断源的。
17. 请给出将定时器0设置成每毫秒产生一次高优先级中断所需的步骤。中断向量地址是多少？
18. 在一个应用中，CPU寄存器已被设置成从外部中断源INT0、INT1和INT2接受中断。假设一个中断已经被检测到。请说明如何确定这个中断源。 117

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com
tyw藏书

第3章 C编程语言

对于PIC18系列微控制器，市场上有几种C编译器。这些编译器有许多相似的特点，并且都能用来开发PIC18微控制器的基于C语言的高级程序。

以下是在商业、工业和教育方面的PIC18微控制器应用中，最常使用的一些C编译器：

- mikroC
- PICC18
- C18
- CCS

由MikroElektronika公司（网址：www.microe.com）开发的mikroC，比较流行而且功能强大，很容易学习，附带有丰富的资源，比如带有一个数量很大的函数库、一个集成有内置仿真器和内电路调试器（如mikroICD）的开发环境。MikroElektronika公司提供程序容量限制为2KB的演示版编译器。

由Hi-Tech软件公司（网址：www.htsoft.com）开发的PICC18，是另一个流行的C编译器，有两种可用的版本：标准版和专业版。该公司还提供一个功能强大的仿真器和一个集成的开发环境（Hi-Tide）。PICC18得到了用于PIC微控制器系统仿真的PROTEUS仿真器（www.labcenter.co.uk）的支持。在开发者的网站上可以找到这个编译器的演示版，不过设定了使用期限。

119

C18是Microchip公司（网址：www.microchip.com）的产品。在Microchip公司的网站上，可以找到一个有使用期限的演示版和一个不限制时间而限制功能的C18。C18包括一个仿真器，并且支持诸如内电路模拟器（如ICE2000）和内电路调试器（如ICD2）这样的硬件和软件开发工具。

CCS是Custom计算机系统公司（网址：www.ccsinfo.com）开发出来的产品。该公司提供带使用期限的CCS编译器。CCS提供大量的内置函数，支持内电路调试器（如ICD-U40），这在开发基于PIC18微控制器的系统时很有帮助。

本书主要讨论mikroC编译器的用法，多数项目均基于该编译器开发。

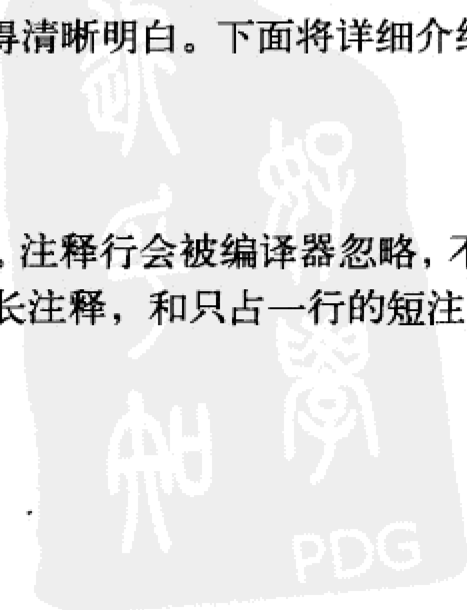
3.1 mikroC程序的结构

120

图3-1所示的是一个最简单的mikroC程序的结构。该程序的功能是让一个连接到PIC微控制器的端口RB0（PORTB的位0）上的LED灯每隔1秒时间闪亮1次。在现阶段，读者不用担心自己不明白该程序的操作，随着本章的深入介绍这将会变得清晰明白。下面将详细介绍图3-1所示的编程要素。

3.1.1 注释

注释用来说明程序的运行或者解释一个程序语句。注释行会被编译器忽略，不会进行编译。在mikroC程序中，注释有两种形式：可以有好几行的长注释，和只占一行的短注释。在程序开



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

头的注释行通常用来简短地描述程序的运行，并提供作者的名字、程序文件名，说明编制程序的时间，列出版本号及每个版本对应的修改内容。如图3-1所示，跟在语句后面的注释可用来描述该语句所执行的操作。清晰简洁的注释行对于一个程序的维护和生命期是很重要的，因为具有很好注释的程序更容易修改和/或者更新。

如图3-1所示，长注释以符号“/*”开始而以符号“*/”结尾，而短注释以符号“//”开始且不需要结尾符号。

```
/*************************************************************************

                    LED FLASHING PROGRAM
                    ************************************

This program flashes an LED connected to port pin RB0 of PORTB with one
second intervals.

 Programmer   : D. Ibrahim
 File         : LED.C
 Date         : May, 2007
 Micro        : PIC18F452
*************************************************************************/

void main()
{
    for(;;)                          // Endless loop
    {
        TRISB = 0;                   // Configure PORTB as output
        PORTB.0 = 0;                 // RB0 = 0
        Delay_Ms(1000);              // Wait 1 second
        PORTB.0 = 1;                 // RB0 = 1
        Delay_Ms(1000);              // Wait 1 second
    }                                // End of loop
}
```

图3-1　一个简单C程序的结构

3.1.2　一个程序的开始和结束

C语言程序以下面的关键字开始：

```
void main()
```

在这个关键字后面，使用一个左花括号来表示程序体的开始。程序以一个右花括号结束。因此，图3-1所示的程序结构如下：

```
void main()
{
            程序体
}
```

3.1.3　程序语句的结尾

在C语言中，所有的程序语句都必须以分号（;）结尾，否则编译器会生成错误：

```
j=5;            //正确
j=5             //错误
```

3.1.4　空白

空白是指空格、空行、制表符和换行符。C编译器会忽略所有的空白。因此，下面的三种

121

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

写法作用是相同的：

```
int i;      char j;
或者
int i;
char j;
或者
int i;
            char j;
```

相似地，下面两种写法也是相同的：

```
i=j+2;
或者
i=j
   +2;
```

3.1.5 区分大小写

一般地，C语言是区分大小写的，并且小写名字的变量和大写名字的变量是不同的。然而，目前mikroC变量是不区分大小写的（尽管将来发布的mikroC可能会区分大小写），因此下面的变量是相同的：

```
total  TOTAL   Total   ToTal   toTal   totaL
```

唯一例外的是标识符main和interrupt，在mikroC中必须使用小写。为了兼容其他的C编译器，这本书中假设变量都是区分大小写的，不使用名字相同而大小写不一样的变量。

122

3.1.6 变量名

在C语言中，变量名可以是以一个字符或者下划线开头。本质上，变量名可以包括从a～z和从A～Z的任意一个字符、数字0～9和下划线符号“_”。每个变量名的前31个字符应该是唯一的。变量名可以包括大写和小写的字符（请参阅3.1.5节），而数字符号可以使用在变量名中间。合法的变量名的例子如下：

```
Sum    count    sum100    counter    i1    UserName
_myname
```

对于编译器，有些名字是保留字，在程序中不能用作变量名。表3-1列出了这些保留字的名字。

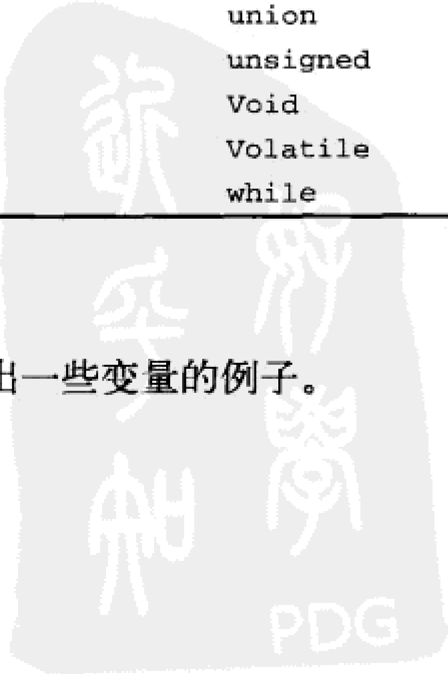
表3-1 mikroC保留字的名字

asm	enum	signed
auto	extern	sizeof
break	float	static
case	for	struct
char	goto	switch
const	if	typedef
continue	int	union
default	long	unsigned
do	register	Void
double	return	Volatile
else	short	while

3.1.7 变量类型

123

mikroC语言支持的变量类型如表3-2所示。这部分将给出一些变量的例子。



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

表3-2 mikroC变量类型

类　　型	字长（位）	取值范围
unsigned char	8	0 to 255
unsigned short int	8	0 to 255
unsigned int	16	0 to 65535
unsigned long int	32	0 to 4294967295
signed char	8	-128 to 127
signed short int	8	-128 to 127
signed int	16	-32768 to 32767
signed long int	32	-2147483648 to 2147483647
float	32	±1.17549435082E-38 to ±6.80564774407E38
double	32	±1.17549435082E-38 to ±6.80564774407E38
long double	32	±1.17549435082E-38 to ±6.80564774407E38

1. (unsigned)char 或者unsigned short(int)类型

(unsigned) char或者unsigned short (int) 类型是8位的无符号变量，其取值范围在0～255之间。在下面的例子中，创建了两个8位的变量total和sum，并且sum被赋值为十进制数150：

```
unsigned char total, num;
sum=150;
```

或者

```
char total, sum;
sum=150;
```

124

在这些变量的声明中，可以对变量进行赋值。因此，上面的语句也可以写作：

```
char total, sum=150;
```

2. signed char或者(signed)short(int)类型

signed char或者 (signed) short (int)，是8位有符号字符变量，其取值范围在-128~+127之间。在下面的例子中，创建了一个名为counter的8位有符号变量，并赋值为-50：

```
signed char counter=-50;
```

或者

```
short counter=-50;
```

或者

```
short int counter=-50;
```

3. (signed)int类型

(signed) int是16位变量，其取值范围在-32 768～+32 767之间。在下面的例子中，一个名为Big的有符号整数被创建：

```
int Big;
```

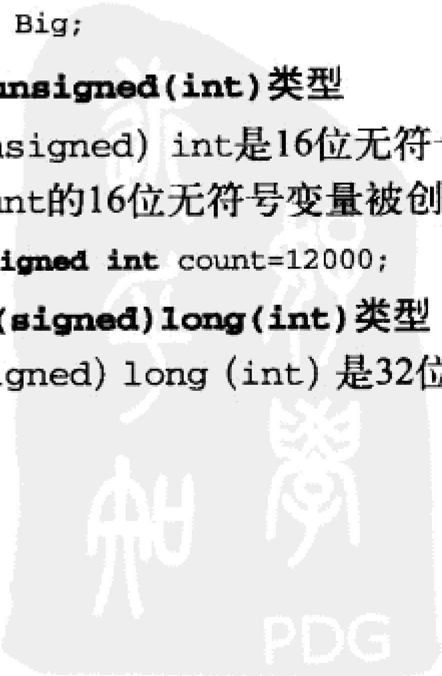
4. unsigned(int)类型

(unsigned) int是16位无符号变量，其取值范围在0~65 535之间。在下面的例子中，一个名为count的16位无符号变量被创建，并被赋值为12 000：

```
unsigned int count=12000;
```

5. (signed)long(int)类型

(signed) long (int) 是32位的长整型变量，其取值范围在-2 147 483 648～+2 147 483 647



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

之间。如下例所示：

```
signed long LargeNumber;
```

6. **unsigned long(int)类型**

(unsigned) long (int) 是32位的无符号变量，其取值范围在0~4 294 967 295之间。如下例所示：

125

```
unsigned long VeryLargeNumber;
```

7. **float或者double或者long double类型**

float 或者double或者long double类型，是在mikroC中实现的浮点型变量，使用Microchip AN575的32位格式，兼容IEEE 754。浮点数的取值范围是从±1.17549435082E-38～±6.80564774407E38。在下面的例子中，一个名字为area浮点型变量被创建，并被赋值为12.235：

```
float area;
area=12.235;
```

在程序开发期间为避免冲突，建议指明变量的符号（有符号或者无符号）和变量的类型。例如，使用unsigned char，而不是仅使用char;使用unsigned int，而不是仅使用unsigned。

在这本书中，使用下面的mikroC数据类型，这些数据类型很容易记忆，且和大多数其他的C编译器是兼容的。

unsigned char	0～255
signed char	-128～127
unsigned int	0～65 535
signed int	-32 768～32 767
unsigned long	0～4 294 967 295
signed long	-2 147 483 648～2 147 483 647
float	±1.17549435082E-38～±6.80564774407E38

3.1.8 常量

常量在程序中表示固定的值（数字或者字符），不能被改变。常量存储在PIC微控制器的闪存中，因此不会浪费宝贵而有限的RAM内存。在mikroC中，常量可以是整数、浮点数、字符、字符串或者枚举类型。

1. **整数型常量**

整数常量可以是十进制、十六进制、八进制或者二进制。一个常量的数据类型可以由编译器根据它的值来获取。但是后缀可以用来改变一个常量的类型。

126

在表3-2中，可以看到十进制常量的值可以是从-2 147 483 648～+4 294 967 295。例如，常数210是作为一个unsigned char（或者unsigned short int）类型来存储的。相似地，常数-200是作为一个signed int类型来存储的。

使用后缀u或者U可以强制性地使常量为unsigned类型。使用后缀L或者l可以强制性地使常量为long类型。同时使用U（或者u）和L（或者l）可以强制性地使常量为unsigned long类型。

常量使用关键字const来声明，并且存储在PIC微控制器的闪存中，因此不会浪费宝贵的RAM空间。在下面的例子中，常量MAX被赋值为100，并且被存储在PIC微控制器的闪存中：

```
const MAX=100;
```

十六进制常量以字符0x或者0X开头，且可以包括数字0～9和十六进制字符A～F。在下面的例子中，常量TOTAL被赋值为十六进制数FF：



世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

tyw藏书

```
const TOTAL=0xFF;
```

八进制常量以数字0作为开头，并且可以包括数字0～7。在下面的例子中，常量CNT被赋值为八进制数17：

```
const CNT=017;
```

二进制常数以0b或者0B开头，并且只可以包括0或1。在下面的例子中，一个名为Min的常量被赋值为二进制数11110000：

```
const Min=0b11110000;
```

2. 浮点型常量

浮点常数包括整数部分、小数点、小数部分和一个可选的e或者E（后面紧跟一个有符号的整数型指数）。在下面的例子中，一个名字为TEMP的常量被赋值为小数37.50：

```
const TEMP=37.50;
```

或者

```
const TEMP=37.50E1
```

127

3. 字符型常量

字符型常量是一个使用单引号标记的字符。在下面的例子中，一个名为First_Alpha的常量被赋值为字符‘A’：

```
const First_Alpha='A'
```

4. 字符串常量

字符串常量是存储在微控制器闪存中的具有固定序列的一串字符。字符串开头和结尾都必须使用双引号字符（“）。编译器会自动的插入一个空符号作为一个字符串的结尾。下面是一个字符串常量的例子：

```
"This is an example string constant"
```

一个字符串常量通过使用一个反斜线符号（“\”）来进行跨行扩展：

```
"This  is  first  part  of  the  string\
and this is the continuation of the string"
```

该字符串常量等同于：

```
"This  is  first  part  of  the  string and this is the continuation of the string"
```

5. 枚举常量

枚举常量是整数类型，常用来使程序更容易理解。在下面的例子中，常量colors包含了各种颜色的名字。第一个元素被赋值为0：

```
enum colors{black,brown,red,orange,yellow,green,blue,gray,white};
```

3.1.9 转义序列

转义序列用来代表不可打印的ASCII码字符。表3-3列出了一些常用的转义序列和它们在C语言中所表示的意思。例如，字符组合“\n”表示换行符。

128

表3-3 一些常用的转义序列

转义序列	十六进制值	字符含义
\a	0x07	响铃符
\b	0x08	回退符

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

（续）

转义序列	十六进制值	字符含义
\t	0x09	水平制表符
\n	0x0A	换行符
\v	0x0B	垂直制表符
\f	0x0C	换页符
\r	0x0D	回车符
\xH		十六进制数字的字符串

一个ASCII码可以表示为在一个反斜线符号之后指定一个十六进制的数。例如，换行符也能表示为“\x0A”。

3.1.10　静态变量

静态变量是用在函数中的局部变量（请参阅第4章），在对函数的连续调用中必须保存静态变量的最新值。如下面的例子所示，静态变量的声明使用关键字static：

```
static unsigned int count;
```

3.1.11　外部变量

外部变量是在变量名前面使用关键字extern来声明的。它告诉编译器这个变量在其他分离的源代码模块中已被声明。在下面的例子中，变量sum1和sum2被声明为外部无符号整数变量：

129

```
extern int sum1, sum2;
```

3.1.12　动态变量

动态变量在基于中断的程序和输入输出子程序中特别重要。使用关键字volatile意味着变量的值在程序的生命期中是可变的，而不依赖于程序的正常执行流。声明为volatile的变量不能由编译器优化，因为变量的值可能会意外地改变。在下面的例子中，变量Led被声明为一个volatile的无符号字符：

```
volatile unsigned char Led;
```

3.1.13　枚举变量

枚举变量的使用可提高程序的可读性。在一个枚举变量中，指定了一列表项，其中第一项的值被设为0，接下来的一项被设为1，依此类推。在下面的例子中，类型Week被声明为一个枚举列表，并且有MON=0、TUE=1、WED=2，等等：

```
enum Week{MON, TUE, WED, THU, FRI, SAT, SUN};
```

在一个列表中指定元素的值是可能的。在下面的例子中，black=2，blue=3，red=4等：

```
enum colors{ black=2,blue,red,white,gray};
```

相似地，在下面的例子中，black=2，blue=3，red=8，gray=9：

```
enum colors{ black=2,blue,red=8, gray};
```

可以在列表后面声明枚举类型的变量。例如，要声明变量My__Week为枚举类型Week，可以使用下面的语句：

```
enum Week{MON, TUE, WED, THU, FRI, SAT, SUN}My__Week;
```

PDG

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

tyw藏书

现在，可以在程序中使用变量My__Week：

```
My__Week=WED              //给My_Week 赋值2
```

或者

```
My__Week=2                //同上
```

130

在定义枚举类型Week之后，可以声明Week类型的变量This__Week和Next__Week如下：

```
enum Week This__Week,Next__Week;
```

3.1.14 数组

数组用来存储在同一块内存中并有共同名字的数据项。通过指定数组的类型、名字和它将存储元素的数量来声明它。例如：

```
unsigned int Total[5];
```

这个无符号整数类型的数组名为Total，有五个元素。数组的第一个元素被标号为0。因此，在这个例子中，Total[0]指数组的第一个元素，而Total[4]指数组的最后一个元素。数组Total是在内存中以五个连续的地址存储，如下：

Total[0]
Total[1]
Total[2]
Total[3]
Total[4]

数据可以通过指定数组的名字和索引号来存储。例如，要将25存储在数组的第2个元素中，则需要写成：

```
Total[1]=25;
```

相似地，一个数组的内容可以通过指定数组的名字和它的索引号来读出。例如，要把第3个数组元素复制给一个名为Temp的变量，则需要写成：

```
Temp=Total[2];
```

数组的内容可以通过在声明数组时分配一个用逗号隔开的序列来初始化。在下面的例子中，数组months有12个元素，并且months[0]=31，months[1]=28，依此类推：

```
unsigned char months[12]={31,28,31,30,31,30,31,31,30,31,30,31};
```

131

在声明数组时也可以不指定它的长度：

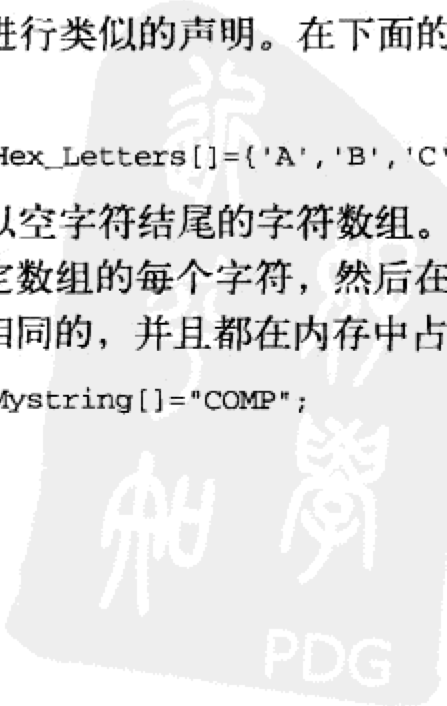
```
unsigned char months[]={31,28,31,30,31,30,31,31,30,31,30,31};
```

字符数组可以进行类似的声明。在下面的例子中，一个名为Hex__Letters有6个元素的字符数组被声明：

```
unsigned char Hex_Letters[]={'A','B','C','D','E','F'};
```

字符串是一个以空字符结尾的字符数组。字符串的声明可以是把字符串括在双引号内，或者是在单引号内指定数组的每个字符，然后在字符串的末尾加上一个空字符。下面例子中的两个字符串的声明是相同的，并且都在内存中占据五个地址：

```
unsigned char Mystring[]="COMP";
```



世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

tyw藏书

和

```
unsigned char Mystring[]={'C','O','M','P','\0'};
```

在C编程语言中，同样可以声明多维数组。一维数组通常叫作向量，二维数组通常叫作矩阵。一个二维数组的声明，需要指定数组的数据类型、数组名字和每一维的长度。在下面的例子中，一个名字为P、有3行4列的二维数组被创建。该数组总共有12个元素。数组的第一个元素是P[0][0]，而最后一个元素是P[2][3]。该数组的结构如下：

P[0][0]	P[0][1]	P[0][2]	P[0][3]
P[1][0]	P[1][1]	P[1][2]	P[1][3]
P[2][0]	P[2][1]	P[2][2]	P[2][3]

一个多维数组的元素可以在声明数组时被指定。在下面的例子中，二维数组Q有2行2列，它的对角线上的元素为1，而非对角线上的元素为0：

132

```
unsigned char Q[2][2]={{1,0},{0,1}};
```

3.1.15　指针

指针在C语言中是一个很重要的部分，因为它们用来保持变量的内存地址。指针的声明和其他变量一样，只是在变量名的前面多了一个字符（*）。通常，指针可以用来指向（或者保持地址）字符变量、整数变量、长变量、浮点变量或者函数（尽管mikroC通常不支持指向函数的指针）。

在下面的例子中，一个名字为pnt的无符号字符指针被声明：

```
unsigned char *pnt;
```

当一个新指针被创建时，它的内容一般未被初始化，也不保持任何变量的地址。可以使用（&）符号来为指针分配一个变量的地址：

```
pnt=&Count;
```

现在，pnt存放着变量Count的地址。可以通过在指针的前面使用符号（*）来设置变量Count的值。例如，使用指针可以将Count赋值为10：

```
*pnt=10;        //Count=10
```

该语句的作用跟下面的语句是一样的。

```
Count=10;       //Count=10
```

或者，可以使用指针将Count的值复制给变量Cnt：

```
Cnt=*pnt;       //Cnt=Count
```

数组指针

在C语言中，数组的名字也是该数组的指针。因此，对于数组：

```
unsigned int Total[10];
```

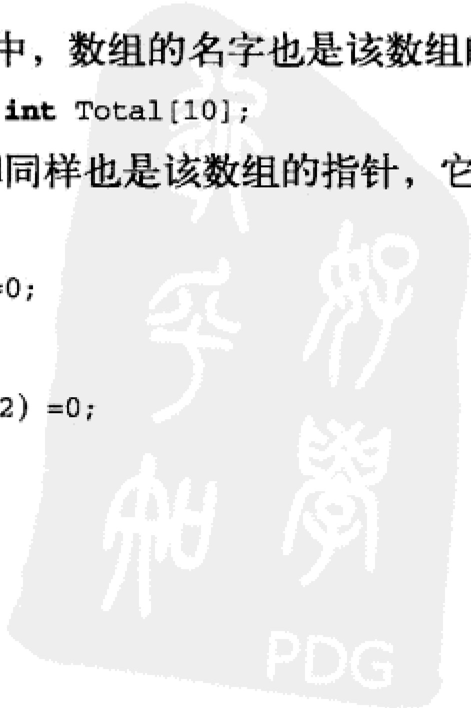
名字Total同样也是该数组的指针，它存放着数组第一个元素的地址。因此下面的两个语句是等效的：

133

```
Total[2]=0;
```

和

```
*(Total+2)=0;
```



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

同样地，下面的语句是正确的：

```
&Total[j]=Total+j;
```

在C语言中，可以执行的指针运算有：

- 比较两个指针；
- 将一个指针加上或者减去一个整数；
- 将两个指针相减；
- 把一个指针赋值给另一个指针；
- 将一个指针同空指针进行比较。

例如，假设指针P存放着数组元素Z[2]的地址：

```
P=&Z[2];
```

现在可以把数组Z的元素2和3清零，正如下面的两个例子所示。除了第一个例子的指针P最后存放着Z[3]的地址，而第二个例子的指针P存放的是Z[2]的地址之外，它们的功能是相同的。

```
*P = 0;              // Z[2] = 0
P = P + 1;           // 指针P存放Z[3]的地址
*P = 0;              // Z[3] = 0
```

或者

```
*P = 0;              // Z[2] = 0
*(P + 1) = 0;        // Z[3] = 0
```

一个指针可以被赋值给另外一个指针。在下面的例子中，使用两个不同的指针将变量Cnt和Tot均设为10：

```
unsigned int *i, *j;     // 声明两个指针
unsigned int Cnt, Tot;   // 声明两个变量
i=&Cnt                   // i指向Cnt
*i=10;                   // Cnt=10
j=i;                     // 把指针i复制给指针j
Tot=*j;                  // Tot=10
```

134

3.1.16 结构体

结构体可以用来集合相关的数据项并把它们当作一个单独的对象。跟数组不一样，结构体可以包含不同的数据类型。例如，一个结构体可以存储一个学生的详细个人信息，如名字、姓、年龄和生日等。

通过使用关键字struct创建，后接一个结构体名字和一个成员声明的列表。可选择在结构体的结尾对结构体变量进行声明。

下面的例子是一个名为Person的结构体的声明：

```
struct Person
{
        unsigned char name[20];
        unsigned char surname[20];
        unsigned char nationality[20];
        unsigned char age;
}
```

声明结构体不会占用任何内存空间，编译器创建一个模板来描述最终存储在结构体变量中的数据对象或者成员元素的名字和类型。只有在相同类型的结构体变量被创建时，这些变量才会占据内存空间。可以通过给出结构体的名字和变量的名字来声明相同类型的结构体变量。例如，Person类型的两个变量Me和You，可以通过下面的语句来创建：

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
struct Person Me, You;
```

135

Person类型的变量还可以在结构体声明中被创建，如下：

```
struct Person
{
        unsigned char name[20];
        unsigned char surname[20];
        unsigned char nationality[20];
        unsigned char age;
} Me, You;
```

可以通过指定结构体的名字，后面接一个圆点（.）和成员的名字来对一个结构体成员进行赋值。在下面的例子中，结构体变量Me的成员age被设为25，并将结构体变量You中age的值赋值给变量M：

```
Me.age=25;
M=You.age;
```

可以在结构体声明的过程中对结构体成员进行初始化。在下面的例子中，结构体变量Cylinder的半径和高度分别被初始化为1.2和2.5：

```
struct Cylinder
{
        float radius;
        float height;
} MyCylinder = {1.2, 2.5};
```

通过定义变量类型为指针，使用指针也可以对结构体成员进行赋值。例如，对于结构体Cylinder，如果定义TheCylinder为一个指针，那么可以编制程序为：

```
struct Cylinder
{
        float radius;
        float height;
} *TheCylinder;

TheCylinder -> radius = 1.2;
TheCylinder -> height = 2.5;
```

结构体的长度是指结构体所包含的字节数。可以使用sizeof运算符来获取结构体的长度。考虑上面的例子，使用语句

```
sizeof (MyCylinder)
```

136

则返回8，因为每个浮点型变量都占据4 B的内存。

可以使用结构体来定义位字段。使用位字段，可以为变量的每一位分配标识符。例如，要将一个变量的位0、位1、位2和位3标识为LowNibble，而将余下的4位标识为HighNibble，则程序可以写为：

```
struct
{
        LowNibble    : 4;
        HighNibble   : 4;
} MyVariable;
```

然后，可以访问变量MyVariable的半字节，如下：

```
MyVariable.LowNibble  = 12;
MyVariable.HighNibble = 8;
```

在C语言中，可以使用typedef语句来创建一个新的变量类型。例如，可以如下创建一个名为Reg的新的结构体数据类型：

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
typedef struct
{
        unsigned char name[20];
        unsigned char surname[20];
        unsigned age;
} Reg;
```

接下来，Reg类型变量的创建方式和其他类型变量创建方式是相同的。在下面的例子中，变量MyReg、Reg1和Reg2被创建为数据类型Reg：

```
Reg MyReg, Reg1, Reg2;
```

假设两个结构体都是从同一个模板派生而来，那么一个结构体的内容是可以被复制到另一个结构体的。在下面的例子中，相同类型的结构体变量P1和P2被创建，并且将P2复制给P1：

```
struct Person
{
        unsigned char name[20];
        unsigned char surname[20];
        unsigned int age;
        unsigned int height;
        unsigned weight;
}
struct Person P1, P2;
........................
........................
P2 = P1;
```

137

3.1.17 联合体

联合体用来重载变量。联合体和结构体很相似，甚至是以相似的方式定义的。它们都是基于模板的，其成员都是使用“.”或者“->”运算符来访问。但是它们也是有区别的，因为联合体中所有的变量都占据相同的内存区域，也就是说，它们共享同一内存空间。以下是一个联合体声明的例子：

```
union flags
{
        unsigned char x;
        unsigned int y;
} P;
```

在这个例子中，变量x和y占据同样的内存区域，并且该联合体的长度为2 B，也是联合体长度的最大值。当变量y被载入一个2 B的值时，变量x和y的低字节地址有相同的值。在下面的例子中，y被载入16位的十六进制数0xAEFA，而x被载入0xFA：

```
P.y=0xAEFA;
```

联合体的长度是指其成员的最大长度（字节数）。因此，使用语句：

```
sizeof (P)
```

返回值为2。

这个联合体也可以声明为：

```
union flags
{
        unsigned char x;
        unsigned int y;
}
union flags P;
```

138

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

3.1.18 C语言的运算符

运算符用于表达式中的变量和其他对象，以引起确定的条件或者计算发生。

MikroC语言支持下面的运算符：

- 算术运算符；
- 关系运算符；
- 逻辑运算符；
- 按位运算符；
- 赋值运算符；
- 条件运算符；
- 预处理程序运算符。

1. 算术运算符

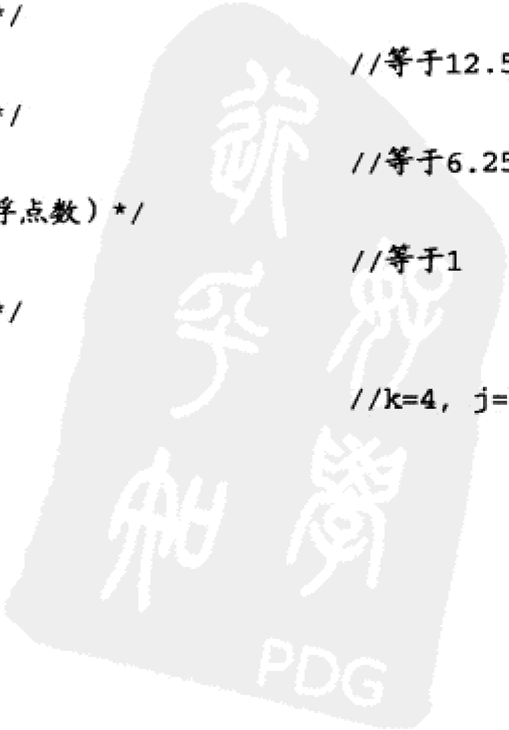
算术运算符用在算术计算中。算术运算符的结合是从左到右的，且返回数字值。mikroC的算术运算符列表如表3-4所示。

表3-4 mikroC的算术运算符

运 算 符	运算操作	运 算 符	运算操作
+	加法	%	余数（整数除法）
—	减法	++	自动增量
*	乘法	——	自动减量
/	除法		

下面的例子说明算术运算符的使用方法：

```
/*两整数相加*/
5+12                    //等于17
/*两整数相减*/
120-5                   //等于115
10-15                   //等于-5
/*两整数相除*/
5/3                     //等于1
12/3                    //等于4
/*两整数相乘*/
3*12                    //等于36
/*两浮点数相加*/
3.1+2.4                 //等于5.5
/*两浮点数相乘*/
2.5*5.0                 //等于12.5
/*两浮点数相除*/
25.0/4.0                //等于6.25
/*求余（不包括浮点数）*/
7%3                     //等于1
/*后自增运算符*/
j=4;
k=j++;                  //k=4, j=5
```



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
/*前自增运算符*/
j=4;
k=++j;                    //k=5, j=5
/*后自减运算符*/
j=12;
k=j--;                    //k=12, j=11
/*前自减运算符*/
j=12;
k=--j;                    //k=11, j=11
```

140

2. 关系运算符

关系运算符用在比较中。如果表达式计算为TRUE，则返回1；否则返回0。

所有关系运算符的结合都是从左到右的。在表3-5中列出了mikroC的关系运算符。

表3-5 mikroC的关系运算符

运算符	运算操作	运算符	运算操作
==	等于	<	小于
!=	不等于	>=	大于或等于
>	大于	<=	小于或等于

下面例子说明了关系运算符的使用方法：

```
x = 10
x > 8       //返回1
x == 10     //返回1
x < 100     //返回1
x > 20      //返回0
x != 10     //返回0
x >= 10     //返回1
x <= 10     //返回1
```

3. 逻辑运算符

141

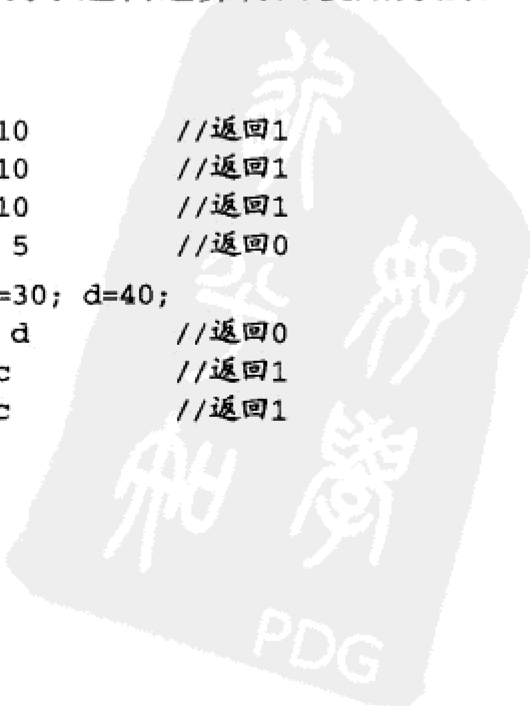
逻辑运算符用在逻辑和算术比较中。如果表达式计算为非零，则返回TRUE（即逻辑1）。如果表达式计算为0，则返回FALSE（即逻辑0）。如果在一个语句中使用多个逻辑运算符，并且第一个条件计算为FALSE，那么将不计算第二个表达式。

mikroC的逻辑运算符如表3-6所示。

表3-6 mikroC的逻辑运算符

运算符	运算操作	运算符	运算操作
&&	逻辑与运算	!	逻辑非运算
||	逻辑或运算		

下面的例子说明了逻辑运算符的使用方法：



```
/*逻辑与*/
x = 7;
x > 0 && x < 10         //返回1
x > 0 || x < 10         //返回1
x >=0 && x <=10         //返回1
x >=0 && x <  5         //返回0
a=10; b=20; c=30; d=40;
  a> b && c > d         //返回0
b > a && d > c          //返回1
a > b || d > c          //返回1
```

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

4. 位运算符

位运算符用来修改变量的位。mikroC的位运算符如表3-7所示。

如果两个位均为1，则执行按位逻辑与运算（AND）将返回1；否则返回0。

如果两个位均为0，则执行按位逻辑或运算（OR）将返回0；否则返回1。

如果两个位的值是相反的，则执行按位逻辑异或运算（XOR）将返回1；否则返回0。

142

按位取反运算，即把每一位取反。

按位左移和按位右移分别将二进制位向左边或向右边移动。

表3-7　mikroC的位运算符

运 算 符	运算操作	运 算 符	运算操作
&	按位逻辑与运算	~	按位取反运算
|	按位逻辑或运算	<<	左移
^	按位逻辑异或运算	>>	右移

下面的例子说明了位运算符的使用方法：

```
i.    0xFA & 0xEE returns 0xEA
      0xFA:     1111 1010
      0xEE:     1110 1110
      -------------
      0xEA:     1110 1010

ii.   0x01 | 0xFE returns 0xFF
      0x08:     0000 0001
      0xFE:     1111 1110
      -------------
      0xFE:     1111 1111

iii.  0xAA ^ 0x1F returns
      0xAA:     1010 1010
      0x1F:     0001 1111
      -------------
      0xB5:     1011 0101

iv.   ~0xAA returns 0x
      0xAA:     1010 1010
      ~ :       0101 0101
      -------------
      0x55:     0101 0101

v.    0x14 >> 1 returns 0x08 (shift 0x14 right by 1 digit)
      0x14:     0001 0100
      >>1 :     0000 1010
      -------------
      0x0A:     0000 1010

vi.   0x14 >> 2 returns 0x05 (shift 0x14 right by 2 digits)
      0x14:     0001 0100
      >> 2:     0000 0101
      -------------
      0x05:     0000 0101

vii.  0x235A << 1 returns 0x46B4 (shift left 0x235A left by 1 digit)
      0x235A:  0010 0011 0101 1010
      <<1 :    0100 0110 1011 0100
      ---------------------
      0x46B4 : 0100 0110 1011 0100
```

143

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
viii. 0x1A << 3 returns 0xD0 (shift left 0x1A by 3 digits)
      0x1A:    0001 1010
      <<3 :    1101 0000
      ------------
      0xD0:    1101 0000
```

5. 赋值运算符

C语言有两类赋值运算：简单赋值运算和复合赋值运算。在简单赋值运算中，一个表达式仅仅是赋值给另一个表达式，或者使用一个表达式执行运算并将得到的结果赋值给另一个表达式：

```
Expression1=Expression2
```

或者

```
Result= Expression1 operation Expression2
```

一个简单赋值运算的例子如下：

```
Temp=10;
Cnt=Cnt+Temp;
```

复合赋值运算有一个通用形式：

```
Result operation =Expression1
```

这里指定的运算是关于Expression1的，而结果存储在Result中。例如：

144

```
j+=k;    等同于:    j=j+k;
```

同样地，

```
p*=m;    等同于:    p=p*m;
```

下面的复合运算符可以在mikroC的程序中使用：

```
+=    -=    *=    /=    %=
&=    |=    ^=    >>=   <<=
```

6. 条件运算符

条件运算符的语法是：

```
Result=Expression1?Expression2: Expression3
```

首先计算Expression1，如果它的值是真，则把Expression2的值赋给Result，否则把Expression3的值赋给Result。在下面的例子中，通过比较x和y来找出x与y的最大值。如果x>y，则max=x；否则max=y：

```
max=(x>y)?x : y;
```

在下面的例子中，将小写字母转换为大写字母。如果字母是小写的（在a～z之间），那么通过减去32可以得到对应的大写字母：

```
c=(c>=a&&c<=z)?(c-32) : c;
```

7. 预处理程序运算符

预处理程序允许程序员执行下列操作：

- 有条件地编译程序，比如部分代码未被编译；
- 使用其他的符号或者值代替原来的符号；
- 在程序中插入文本文件。

145

预处理程序运算符为（#）字符，以（#）开头的任意一行程序代码都被假定为一个预处理程序命令。这里不需要使用分号字符（;）来终止一个预处理程序命令。

mikroC编译器支持以下的预处理程序命令：

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
#define    #undef
#if        #elif      #endif
#ifdef     #ifndef
#error
#line
```

#define，#undef，#ifdef，#ifndef　# define预处理程序命令用来提供宏扩展，使程序中每次出现的标识符都会被标识符的值所代替。例如，要使用值100来代替MAX的每次出现，可以写为：

```
# define MAX 100
```

已经被定义的标识符不能被重复定义，除非两次定义的值是一样的。解决这个问题的一个办法是删除宏定义：

```
# undef  MAX
```

不然将检测到一个已经存在宏定义。在下面的例子中，如果MAX还没有被定义，则将它赋值为100，否则#define行将被跳过：

```
# ifndef  MAX
       # define  MAX=100
# endif
```

注意，# define预处理程序命令不会占据任何内存空间。

可以通过在宏名字后面的一个括号内指定参数来将参数传递给宏定义。例如，考虑下面的宏定义：

```
# define ADD (a, b)      (a+b)
```

在程序中使用这个宏后，（a，b）会被（a+b）所代替，如下所示：

146

```
p=ADD (x, y)     将会转化为  p= (x+y)
```

相似地，可以定义一个宏来计算两个数的平方值：

```
#define SQUARE (a)         (a*a)
```

现在可以在程序中使用这个宏：

```
p = SQUARE (x)     将被转化为p= (x*x)
```

include　预处理程序指令#include用来在程序中包含一个源文件。通常，带有扩展名“.h”的头文件需要使用# include。#include的有两种使用形式：

```
#include <file>
```

和

```
# include "file"
```

在第一种形式中，首先在mikroC的安装目录搜索文件，然后才在用户搜索路径中搜索。在第二种形式中，指定的文件首先在mikroC的项目文件夹中搜索，然后在mikroC的安装文件夹搜索，最后在用户搜索路径中搜索。也可以指定一个完整的目录路径，如下所示：

```
#include "C: \temp\last.h"
```

文件的搜索范围仅限于指定的目录路径。

#if，# elif，# else，#endif　预处理程序命令#if、# elif、# else和#endif是用来进行条件编译的，仅当满足一定的条件时才编译其中的部分源代码。在下面的例子中，如果M是一个非零值，则其中变量A和B被清零的代码部分将被编译；否则其中A和B均被置1的代码部分将被编译。注意，#if必须以#endif结尾：

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
#if M
        A = 0;
        B = 0;
#else
        A = 1;
        B = 1;
#endif
```

147

如果以前的条件是错的，也可以使用#elif条件来测试一个新条件：

```
#if M
        A = 0;
        B = 0;
#elif N
        A = 1;
        B = 1;
#else
        A = 2;
        B = 2;
#endif
```

在上面的例子中，如果M是一个非零值，则程序段A=0; B=0;将被编译。否则，如果N是个非零值，那么程序段A=1; B=1;将被编译。最后，如果M和N都为0，那么程序段A=2; B=2;将被编译。注意，在#if 和#endif之间仅有一个程序段被编译，并且该程序段可以包括任意数量的语句。

3.1.19 修改控制流

在正常情况下，程序语句通常按照从头到尾的顺序来执行。在一个C语言程序中，还可以使用控制语句来修改正常的顺序流。下面的控制语句在mikroC程序中是有效的：

- 选择语句；
- 无条件修改控制流；
- 迭代语句。

1. 选择语句

这里，有两个选择语句：if和switch。

if语句 if语句的一般形式是：

```
if (表达式)
    语句1;
else
    语句2;
```

148

或者

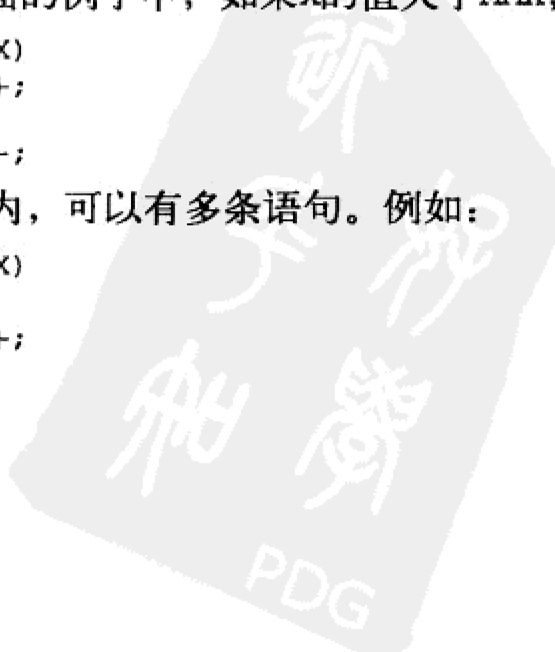
```
if (表达式) 语句1;else 语句2;
```

如果表达式计算值为TRUE，则执行语句1；否则执行语句2。关键字else是可选择的，可以被省略。在下面的例子中，如果x的值大于MAX，那么变量p就增加1；否则就减1：

```
if(x > MAX)
        P++;
else
        P--;
```

在花括号内，可以有多条语句。例如：

```
if(x > MAX)
{
        P++;
```



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
        Cnt = P;
        Sum = Sum + Cnt;
}
else
        P--;
```

在这个例子中，如果x的值大于MAX，那么执行花括号内的三条语句；否则执行语句P--。
另一个使用if语句的例子如下：

```
if(x > 0 && x < 10)
{
        Total += Sum;
        Sum++;
}
else
{
        Total = 0;
        Sum = 0;
}
```

149

switch语句 当有很多的条件且只有一个条件为真时，可以使用switch语句来执行不同的操作。switch语句的语法是：

```
switch (条件)
{
      case 条件1:
            语句;
            break;
      case 条件2:
            语句;
            break;
      .....................
      .....................
      case 条件N:
            语句;
            break;
      default:
            语句;
}
```

switch语句的功能如下：首先计算条件，然后和条件1进行比较。如果相匹配，则执行该条件对应的语句块，当遇到关键字break时控制跳出switch语句；如果不匹配，则同条件2进行比较，如果匹配，则执行该条件对应的语句块，然后跳出switch语句；依此类推。default是可选的，如果条件与关键字case后面指定的任意一个条件都不匹配，则执行default后面的语句。

在下面的例子中，变量Cnt的值被计算。如果Cnt=1，则A被置1；如果Cnt=10，则B被置1；如果Cnt=100，则C被置1；如果Cnt不等于1、10或100，那么D被置1：

```
switch (Cnt)
{
        case 1:
                A = 1;
                break;
        case 10:
                B = 1;
                break;
        case 100:
                C = 1;
                break;
        default:
                D = 1;
}
```

150

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

因为在C语言中空白是被忽略的，所以也可以把前面的代码写成：

```
switch (Cnt)
{
        case 1:     A = 1;    break;
        case 10:    B = 1;    break;
        case 100:   C = 1;    break;
        default:    D = 1;
}
```

例3.1 在一个实验中，得到X与Y的值之间的关系为：

```
X    Y
1    3.2
2    2.5
3    8.9
4    1.2
5    12.9
```

编制一个switch语句，给出X的值，返回Y的值。

解 所要求的switch语句为：

```
switch (X)
{
        case 1:
                    Y = 3.2;
                    break;
        case 2:
                    Y = 2.5;
                    break;
        case 3:
                    Y = 8.9;
                    break;
        case 4:
                    Y = 1.2;
                    break;
        case 5:
                    Y = 12.9;
}
```

151

2. 迭代语句

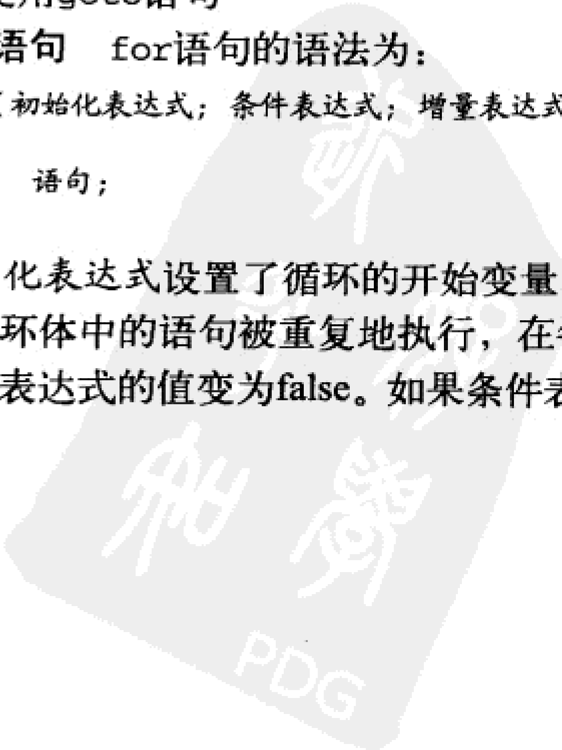
迭代语句允许在程序中执行循环操作，其中循环代码必须被重复执行若干次。在mikroC中，执行迭代可以有4种方式。下面举例介绍每一种方式：

- 使用for语句
- 使用while语句
- 使用do语句
- 使用goto语句

for语句 for语句的语法为：

```
for (初始化表达式; 条件表达式; 增量表达式)
{
        语句;
}
```

初始化表达式设置了循环的开始变量，在进入循环之前要将该开始变量和条件表达式进行比较。循环体中的语句被重复地执行，在每一次迭代之后增量表达式的值将会增加。迭代将持续到条件表达式的值变为false。如果条件表达式的值总是true，则将会形成一个无限循环。下面



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

的例子所示的是如何设置一个执行10次的循环。在这个例子中，变量i从0开始，且在每次迭代后增加1。当i=10时，在这种情况下条件i<10变为false，循环结束。在从循环中退出时，变量i的值是10：

152

```
for (i=0;i<10;i++)
{
      语句;
}
```

该循环也可以由非零值的初始化表达式开始。这里，变量i从1开始并且当i=11时循环结束。因此，在退出循环的时候，变量i的值为11：

```
for (i=1;i<=10;i++)
{
      语句;
}
```

for循环的参数全部是可选的，且可以省略。如果条件表达式被漏掉了，则编译器会假设它的值是true。在下面的例子中，就形成了一个无限循环，其中条件表达式的值总是true，并且变量i的值从0开始，每次迭代后都会增加：

```
/*变量i不断增量的无限循环*/
for (i=0;; i++)
{
      语句;
}
```

在下面的无限循环例子中，所有的参数都被省略：

```
/*无限循环的例子*/
for (;;)
{
      语句;
}
```

在下面的无限循环中，变量i从1开始并且在循环内部没有增量：

```
/*变量i=1的无限循环*/
for (i=1;;)
{
      语句;
}
```

153

如果在for循环内部只有一个语句，花括号可以被省略，如下所示：

```
for (k=0;k<10;k++)      Total=Total+Sum;
```

也可以使用嵌套的for循环。在一个嵌套for循环中，对于外部循环的每次迭代，都会执行内部的循环。在下面的例子中，内部循环迭代5次，外部循环迭代10次。因此，嵌套循环的总迭代次数为50次：

```
/*嵌套循环的例子*/
for(i = 0; i < 10; i++)
{
        for(j = 0; j < 5; j++)
        {
             语句;
        }
}
```

在下面的例子中，计算一个3×4的矩阵M的所有元素之和，并将结果存储在一个名为Sum的变量中：

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

tyw藏书

```
/*将一个3×4的矩阵中所有元素相加*/
Sum = 0;
for(i = 0; i < 3; i++)
{
        for(j = 0; j < 4; j++)
        {
            Sum = Sum + M[i][j];
        }
}
```

因为只有一个语句被执行，所以前面的例子也可以写成：

```
/*将一个3×4的矩阵中所有元素相加*/
Sum = 0;
for(i = 0; i < 3; i++)
{
        for(j = 0; j < 4; j++) Sum = Sum + M[i][j];
}
```

154

while语句 while语句的语法为：

```
while(条件)
{
        语句;
}
```

这里，语句被重复地执行，直到条件变成false。如果条件在开始进入循环时就为false，那么这个循环将不会执行，且程序将从while循环的结尾处继续执行。在循环中改变条件是很重要的，否则会形成无限循环。

下面的代码说明了怎样使用while语句来创建一个执行10次的循环：

```
/*一个执行10次的循环*/
k = 0;
while (k < 10)
{
        语句;
        k++;
}
```

在代码的开头，变量k为0。因为k小于10，所以开始执行while循环。在循环内，每次迭代后k值就增加1。只要k<10，循环就会重复。当k=10时，循环结束。在循环的结尾处，k的值为10。

注意，如果变量k的值在循环内不增加，则会形成一个无限的循环：

```
/*一个无限循环*/
k = 0;
while (k < 10)
{
            语句;
}
```

将条件的值设置为总是true，可以形成一个无限的循环：

```
/*一个无限循环*/
while (k = k)
{
        语句;
}
```

155

下例计算1～10的数字和，并把结果存储到一个名为sum的变量中：

```
/*计算1~10的数字和*/
unsigned int k, sum;
```

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
k = 1;
sum = 0;
while(k <= 10)
{
        sum = sum + k;
        k++;
}
```

while语句也可以没有循环体。这样的语句是很有用的，例如，如果正在等待一个输入端口改变它的值。在下面的例子中，只要PORTB的位0（PORTB.0）的值为逻辑0，程序将一直等待。当端口引脚变为逻辑1时，程序将继续：

```
while(PORTB.0==0);          //一直等到PORTB.0变为1
```

或者

```
while(PORTB.0);
```

同样地，也可以使用嵌套while语句。

do语句　do语句和while语句很相似，除了该语句的循环一直执行到条件变为false（即只要条件为true循环就执行）之外。do语句的条件是在循环的末尾进行检测的。do语句的语法为：

```
do
{
       语句;
} while(条件);
```

不论条件是true还是false，总是会执行第一次迭代的。这是do语句和while语句之间主要的差别。

156

下面的例子说明了如何使用do语句创建一个执行10次的循环：

```
/*执行10次*/
k=0;
do
{
       语句;
       k++;
} while (k < 10);
```

循环以k=0开始，每次迭代后k的值在循环内部增加。在循环的末尾，k被检测，如果k不小于10，则循环结束。在这个例子中，因为在循环开始时k=0，所以在循环的结束时k的值为10。

如下面的例子所示，如果条件在循环内没有被修改，则会形成一个无限的循环。在这里k总是小于10：

```
/*一个无限循环*/
k=0;
do
{
       语句;
} while (k < 10);
```

如果条件的值被设置为总是true，也会形成一个无限的循环：

```
/*一个无限循环*/
do
{
       语句;
} while (k = k);
```

同样地，也可以使用嵌套do循环。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

3. 无条件修改控制流

goto语句 在程序中，goto语句可以用来改变正常的控制流。它将使得程序跳转到一个指定的标签。标签可以是一个以字母开头、冒号（:）结尾的任意字符组合。

在程序中，可以同时使用goto语句和if语句来创建迭代，虽然不推荐这么使用。下面的例子显示了如何使用goto语句和if语句来创建一个执行10次的循环：

157

```
/*执行10次*/
k = 0;
Loop:
          语句;
      k++;
if(k < 10)goto Loop;
```

在循环开始的时候，循环从标签Loop处开始且变量k=0。在循环内部，执行语句，且k增加1。然后将k的值与10进行比较，如果k<10，则程序跳回到标签Loop。因此，直到条件在最后变成false时，循环总共被执行了10次。在循环结束时，k的值为10。

continue和break语句 continue和break语句可以使用在循环内部，用来修改控制流。continue语句通常和if语句一起使用，可以使得循环跳过一次迭代。下面的例子用来计算除5之外的、从1到10的数字和：

```
/*计算数字1、2、3、4、6、7、8、9、10的和*/
 Sum = 0;
 i = 1;
 for(i = 1; i <= 10; i++)
 {
        if(i == 5) continue;     // 跳过数字5
        Sum = Sum + i;
 }
```

相似地，break语句可以用来从循环内部结束循环。在下面的例子中，只计算了从1到5 的数字和，尽管循环的参数设置为迭代10次：

```
/*计算数字1、2、3、4、5的和*/
Sum = 0;
i = 1;
for(i = 1; i <= 10; i++)
{
       if(i > 5) break;     // 则停止循环i>5
       Sum = Sum + i;
}
```

158

3.1.20 结合 mikroC 和汇编语言

有些时候，把PIC微控制器的汇编语言和mikroC语言结合起来是很有必要的。例如，非常精确的程序延迟可以使用汇编语言来生成。尽管关于汇编语言的知识已超出了本书的范围，但是针对那些熟悉PIC微控制器汇编语言的读者，这里还是讨论了一些如何在mikroC语言中使用汇编语言指令的方法。

可以使用关键字asm（或者_asm，或者__asm）将汇编语言指令包含在一个mikroC程序中。一组汇编指令或者一个单独的指令可以被包含在一对花括号内。其语法为：

```
asm
{
      汇编指令
}
```

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

虽然不允许使用汇编语言格式的注释（以分号字符开始的行），但是mikroC允许在汇编语言程序中使用C语言格式的注释：

```
asm
{
        /*这个汇编代码介绍一个程序的延时*/
        MOVLW 6              //把6载入W
        ........................
        ........................
}
```

虽然可以在汇编语言子程序中使用用户声明的C变量，但是在使用之前对其进行声明和初始化。例如，可以初始化C变量Temp，然后将其载入到W寄存器：

```
unsigned char Temp = 10;
asm
{
        MOVLW Temp               // W = Temp = 10
        ..................
        ..................
}
```

159

诸如预定义的端口名字和寄存器名字之类的全局符号无需初始化就可以在汇编语言子程序中使用。

```
asm
{
        MOVWF PORTB
        .....................
        .....................
}
```

3.2 PIC微控制器输入输出端口编程

根据所使用的微控制器的类型，PIC微控制器输入输出端口被命名为PORTA、PORTB、PORTC，依此类推。端口引脚可以是模拟或者数字模式。在模拟模式下，端口仅用作输入，并需要使用到内置的模数转换器和多路复用电路。在数字模式下，端口引脚可以配置成输入或者输出。TRIS寄存器用来控制端口的方向，每个端口都有一个TRIS寄存器，分别被命名为TRISA、TRISB、TRISC，依此类推。可以通过将TRIS寄存器的一个位清零，将对应的端口位设置为输出模式。相似地，通过将TRIS寄存器的一个位置为1，将对应的端口位设置为输入模式。

可以将端口作为一个8位寄存器来访问，也可以单独访问端口的某一个位。在下面的例子中，PORTB被配置成一个输出端口，且所有的位都被置为1：

```
TRISB=0;             //将PORTB设置为输出
PORTB=0xFF;          //将PORTB的位设置为1
```

相似地，下面的例子显示了怎样将PORTC的高4位设为输入，而将PORTC的低4位设为输出：

```
TRISC=0xF0;
```

输入输出端口的某个端口位可以通过指定期望的位号来访问。在下面的例子中，将PORTB的位2加载给变量P2：

```
P2=PORTB.2;
```

160

端口的所有位可以通过下面语句来取反：

```
PORTB=~PORTB;
```

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

3.3 程序例题

在这部分，将给出一些简单的程序例子，以期读者更加熟悉C语言编程。

例3.2 编制一个程序，将PORTB的所有8个端口引脚设置为逻辑1。

解 将PORTB配置为一个输出端口，然后通过发送十六进制数0xFF，使得所有端口引脚被设置为逻辑1：

```
void main()
{
        TRISB=0;                //配置PORTB为输出
        PORTB=0xFF ;            //设置所有端口引脚为逻辑1
}
```

例3.3 编制一个程序，将PORTB的奇数端口引脚（位1、3、5和7）设置为逻辑1。

解 发送位模式10101010到端口，可以将奇数端口引脚设置为逻辑1。该位模式的十六进制数是0xAA，所求的程序为：

```
void main()
{
        TRISB=0;                //配置PORTB为输出
        PORTB=0xAA ;            //打开奇数端口引脚
}
```

例3.4 编制一个程序，以二进制连续地计数，且将计数值发送到PORTB。从而，PORTB所需的二进制数据为：

161

```
00000000
00000001
00000010
00000011
............
............
11111110
11111111
00000000
............
```

解 使用for循环来创建一个无限的循环，且在循环程序内部将一个变量的值增加，然后发送到PORTB：

```
void main()
{
        unsigned char Cnt=0;
        for(;;)                 //无限循环
        {
            PORTB=Cnt;          //发送Cnt到PORTB
            Cnt++;              //增加Cnt
        }
}
```

例3.5 编制一个程序，将PORTB的所有位先设置为逻辑1，然后设置为逻辑0，这样重复10次。

解 使用for语句来创建一个重复运行10次的循环：

```
void main()
{
        unsigned char j;
```

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
        for (j=0;j<10;j++)          //重复10次
        {
        PORTB=0xFF;                 //设置PORTB引脚为1

        PORTB=0;                    //清零PORTB引脚
        }
}
```

162

例3.6　一个圆柱的半径和高分别为2.5 cm和10 cm。编制一个程序，计算该圆柱的体积。

解　要求编制的程序为：

```
void main()
{
        float Radius=2.5, Height=10;
        float Volume;
        Volume=PI* Radius* Radius* Height;
}
```

例3.7　编制一个程序，在一个具有10个元素的整数数组中找出最大的元素。

解　首先，将变量m设为数组中的第一个元素。然后建立一个循环来找出数组中最大的元素：

```
void main()
{
        unsigned char j;
        int m, A[10];
        m=A[0];     //数组的第一个元素
        for (j=0;j<10;j++)
        {
                if (A[j]>m) m=A[j];
        }
}
```

163

例3.8　编制一个程序，使用while语句来清零一个整数数组M所有的10个元素。

解　如下面的程序所示，变量NUM被定义为10而变量j用作循环计数器：

```
#define NUM 10
void main()
{
        int M[NUM];
        unsigned char j = 0;
        while (j < NUM)
        {
                M[j] = 0;
                j++;
        }
}
```

例3.9　编制一个程序，从0℃开始，把温度从℃值转换为℉值，步长为1℃，最高温度值为100℃，并把结果存储到数组F中。

解　假定给出的温度单位为℃，等效的单位为℉的温度值可使用下面的公式来计算：

$$F=1.8C+32.0$$

可以使用一个for循环来计算以℉为单位的温度值，并把结果存储到数组F中。

```
void main ()
{
        float F[100];
        unsigned char C;
        for(C = 0; C <= 100; C++)
```

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
        {
            F[c]=1.8c+32.0
        }
    }
```

164

3.4 小结

有许多汇编和高级语言是专门提供给PIC18系列微控制器的。本书只介绍mikroC编译器，因为它很容易学习，并且提供一个免费的演示版本，允许用户开发程序长度达到2 KB。

本章介绍了mikroC语言。一个C语言程序可能包含大量的函数和变量，外加一个主程序。主程序的开始由语句void main()来指明。

变量用来存储在计算过程中使用到的数值。在C语言中，所有变量都必须在使用之前被声明。变量可以是一个8位字符、一个16位整数、一个32位长整数或者一个浮点数。因为常量被存储在PIC微控制器的闪存中，所以使用常量可以避免使用宝贵而有限的RAM内存。

本章还介绍了各种控制流和迭代语句，如if语句、switch语句、while语句、do语句和break语句等，并分别给出了一些实例。

指针用来存储变量的地址。在第4章中将看到，指针可以用来在函数和其调用点之间传送信息。例如，指针可以用来在主程序和函数之间传递变量。

3.5 练习题

1. 编制一个C程序，将PORTC的位0和位7设置为逻辑1。
2. 编制一个C程序，连续地倒计数，并把计数值发送到PORTB。
3. 编制一个C程序，将一个具有10个元素的数组的每一个元素都乘以2。
4. 编制一个C程序，将两个矩阵P和Q相加。假设两个矩阵的维数都是3×3，把结果存储在矩阵W中。
5. 重复练习题4，但是这里将矩阵P和Q相乘，并把结果存储在矩阵R中。
6. 请区别变量和常量这两个术语的含义。

165

7. 程序重复是什么含义？请描述在C语言中的while、do-while和for循环操作。
8. 什么是数组？请写出示例语句来定义下面的数组：
   a）一个具有10个整数的数组；
   b）一个具有30个浮点数的数组；
   c）一个具有6行10列的二维数组。
9. 跟踪下面循环的运行。在每一次循环结束，变量z的值将会是什么？

   a）
   ```
   unsigned char j = 0, z = 0;
   while(j < 10)
     {
        z++;
           j++;
     }
   ```
   b）
   ```
   unsigned char z = 10;
   for(j = 0; j < 10; j++)z--;
   ```
10. 给出下面变量的定义，并列出下面的条件测试结果（"true"或"false"）：
    ```
    unsigned int a = 10, b = 2;
    if(a > 10)
    if(b >= 2)
    if(a == 10)
    if(a > 0)
    ```

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

11. 编制一个程序，判断一个数是奇数还是偶数。
12. 分别使用AND、OR和EXOR进行按位操作，确定运算结果。

```
Operand 1:    00010001
Operand 2:    11110001
```

13. 试确定下面每个循环的迭代次数，以及在每种情形下变量j的最终值。

```
a)  for(j = 0;  j < 5; j++)
b)  for(j = 1;  j < 10; j++)
c)  for(j = 0;  j <= 10; j++)
d)  for(j = 0;  j <= 10; j += 2)
e)  for(j = 10; j > 0; j -= 2)
```

166

14. 编制一个程序，计算从1到100的所有正整数的和。
15. 编制一个程序，计算阶乘n，其中0！和1！的计算值为1，$n!=n\times(n-1)!$。
16. 编制一个程序，计算存储在数组中的元素的平均值。假设数组名为M，且有20个元素。
17. 修改练习题16中的程序，找出数组的最小值和最大值。将最小值存储在变量Sml中，而将最大值存储在变量Lrg中。
18. 对于下面的测试条件，请给出等效的if-else语句。

```
a)  (a > b) ? 0 : 1
b)  (x < y) ? (a > b) : (c > d)
```

19. 假设f1和f2均为浮点型变量，请解释为什么下面的控制while循环的测试表达式可能不安全：

```
do
{
        ...............
        ...............
} while(f1 != f2);
```

当f1和f2都是整数时，为什么就不会出现问题呢？你将怎样改进这个while循环呢？

20. 请说出下面的while循环实现的功能。

```
k = 0;
Total = 0;
while (k < 10)
{
        Sum++;
        Total += Sum;
}
```

21. 请说出下面的for循环实现的功能。

```
Cnt = 0;
for(;;)
{
        Cnt++;
}
```

167



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

第 4 章　mikroC 的函数和库

4.1　mikroC 函数

函数是用来执行某一特定功能的独立代码。当必须在主程序的不同部分执行独立操作时，通常都需要创建函数。此外，把一个大程序分割成若干更小的独立函数是一个良好的编程习惯。通过调用函数，可以执行函数内部的语句。

定义函数的一般语法如图4-1所示。数据类型用以指明函数返回数据的类型，其后是函数名称和用括号声明的一组参数，参数之间用逗号分隔。下面花括号中的是函数体，即函数的执行代码。

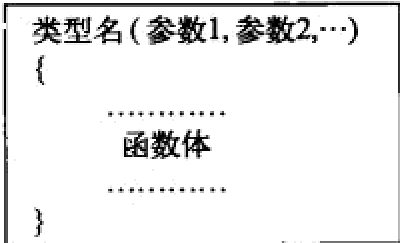


图4-1　函数定义的语法概要

下面是个函数定义的简单例子，函数名为Mult，两个整型变量为a和b，函数返回它们的乘积。注意，在返回语句里，括号是可选的：

```
int Mult (int a, int b)
{
        return (a*b);
}
```

调用函数时，一般应该给出函数的参数列表里描述过的参数。例如，前面的函数可以这样调用：

```
z = Mult (x, y);
```


其中，变量z的数据类型为int。请注意，在函数里已声明过的参数和当函数被调用时传递的参数是相互独立的，即使它们名字相同。在前面的例子里，当函数被调用时，在Mult函数的入口处，变量x将被复制给a，变量y将被复制给b。

有些函数并没有返回值。在这类函数中，数据类型必须声明为void。例如：

```
void LED(unsigned char D)
{
      PORTB=D;
}
```

调用void函数可以没有任何赋值语句,但是必须有括号,括号告诉编译器当前在做函数调用：

```
LED ( );
```

还有一些函数是没有任何参数的。在下面的例子里，函数Compl用来将微控制器的PORTC端口引脚值取反。这个函数没有返回值也没有参数。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
void Compl ( )
{
      PORTC= ~PORTC;
}
```

该函数可以这样调用：

```
Compl ( );
```

通常函数定义在主程序开始之前。

下面将用一些例子来讲述函数定义及其在主程序中的使用。

170

例4.1　编写一个名字Circle_Area的函数，计算圆的面积，将半径作为一个参数。在主程序中使用这个函数来计算一个半径为2.5 cm的圆的面积，并将结果存储在变量Circ里。

解　将函数的数据类型声明为float（浮点型）。圆的面积的计算公式为：

$$面积 = \pi r^2$$

其中，r为圆的半径。计算圆的面积，并将结果存储在变量s（也就是函数的返回值）里。

```
float Circle_Area(float radius)

{
        float s;

        s = PI * radius * radius;
        return s;
}
```

图4-2表明了如何在主函数里使用函数Circle_Area来计算一个半径为2.5 cm的圆的面积。函数的定义在主函数之前。在主函数里，该函数被调用来计算面积，并将结果存储在变量Circ中。

```
/*****************************************************************
                    AREA OF A CIRCLE
                    ================

This program calls to function Circle_Area to calculate the area of a circle.


Programmer:  Dogan Ibrahim
File:        CIRCLE.C
Date:        May, 2007

*****************************************************************/

/* This function calculates the area of a circle given the radius */
float Circle_Area(float radius)
{
        float s;

        s = PI * radius * radius;
        return s;
}

/* Start of main program. Calculate the area of a circle where radius = 2.5 */
void main()
{
        float r, Circ;


        r = 2.5;
        Circ = Circle_Area(r);
}
```

图4-2　计算圆的面积的程序

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

例4.2 编写两个函数Area和Volume，分别用来计算圆柱体的侧面积和体积。然后编写一个主函数来计算一个半径为2.0 cm、高为5.0 cm的圆柱体的侧面积和体积。把计算所得的面积和体积分别存储在变量cyl_area里，把计算所得的体积存储在变量cyl_volume里。

解 计算圆柱体的侧面积的公式如下：

$$面积 = 2\pi rh$$

其中，r和h分别是半径和高。而计算圆柱体的体积的公式如下：

$$体积 = \pi r^2 h$$

图4-3给出了计算圆柱体侧面积和体积的函数。

图4-4给出了计算一个半径为2.0 cm、高为5.0 cm的圆柱体的侧面积和体积的主函数。

```
float Area(float radius, float height)
{
        float s;

        s = 2.0*PI * radius*height;
        return s;
}


float Volume(float radius, float height)
{
        float s;

        s = PI *radius*radius*height;
        return s;
}
```

图4-3 计算圆柱体侧面积和体积的函数

171

```
/*******************************************************************************

                    AREA AND VOLUME OF A CYLINDER
                    =============================

This program calculates the area and volume of a cylinder whose radius is 2.0cm
and height is 5.0cm.

Programmer:   Dogan Ibrahim
File:         CYLINDER.C
Date:         May, 2007
********************************************************************************/

/* Function to calculate the area of a cylinder */
float Area(float radius, float height)
{
        float s;

        s = 2.0*PI * radius*height;
        return s;
}


/* Function to calculate the volume of a cylinder */
float Volume(float radius, float height)
{
        float s;

        s = PI *radius*radius*height;
        return s;
}

/* Start of the main program */
void main()
{
        float r = 2.0, h = 5.0;
        float cyl_area, cyl_volume;

        cyl_area = Area(r, h);
        cyl_volume(r, h);
}
```

图4-4 计算圆柱体的侧面积和体积的程序

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

例4.3 编写一个函数LowerToUpper，用来把小写字母转换成大写字母。

172

解 第一个大写字母A的ASCII码是0x41。类似地，第一个小写字母a的ASCII码是0x61。所以将字母的ASCII码减去0x20，就可以实现从一个大写字母到一个小写字母的转换。图4-5给出了实现上述要求的函数。

```
unsigned char LowerToUpper(unsigned char c)
{
        if(c >= 'a' && c <= 'z')
                return (c – 0x20);
        else
                return c;
}
```

图4-5 将小写字母转换为大写字母的函数

例4.4 在主函数里使用例4.3中编写的函数，把字母r转换成大写字母。

解 满足题目要求的程序如图4-6所示。函数LowerToUpper把变量Lc里的小写字母转换成大写字母，并存放在变量Uc中。

```
/************************************************************************

                LOWERCASE TO UPPERCASE
                ==========================

This program converts the lowercase character in variable Lc to uppercase
and stores in variable Uc.


Programmer:  Dogan Ibrahim
File:        LTOUPPER.C
Date:        May, 2007
************************************************************************/

/* Function to convert a lower case character to upper case */
unsigned char LowerToUpper(unsigned char c)
{
        if(c >= 'a' && c <= 'z')
                return (c – 0x20);
        else
                return c;
}


/* Start of main program */
void main()
{
        unsigned char Lc, Uc;

        Lc = 'r';
        Uc = LowerToUpper(Lc);
}
```

图4-6 调用函数LowerToUpper的程序

4.1.1 函数原型

如果一个函数在调用之前没有被声明，那么编译器就会产生错误信息。解决这个问题的一种方法是创建函数原型。创建函数原型是很容易的，只要复制一个函数头并加上分号就可以了。如果函数有参数，那么参数名称并不是必需的，但定义参数的数值类型是必需的。在下面的例子里，声明了一个名字为Area的函数原型，并给出了参数的类型为float：

173

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com


```
float Area (float radius);
```

这个函数原型也可以声明为：

174

```
float Area (float);
```

函数原型应该在程序的开头部分进行声明。这样，函数的定义和调用就可以在程序的其他任何部分进行。

175

例4.5 重复例4.4，但是将LowerToUpper声明为一个函数原型。

解 如图4-7所示，在程序的开始处声明了LowerToUpper的函数原型。在这个例子里，实际的函数定义部分在主程序之后。

```
/*************************************************************************

                 LOWERCASE TO UPPERCASE
                 ========================

This program converts the lowercase character in variable Lc to uppercase
and stores in variable Uc.


Programmer:  Dogan Ibrahim
File:        LTOUPPER2.C
Date:        May, 2007
*************************************************************************/

unsigned char LowerToUpper(unsigned char);


/* Start of main program */
void main()
{
        unsigned char Lc, Uc;

        Lc = 'r';
        Uc = LowerToUpper(Lc);
}


/* Function to convert a lower case character to upper case */
unsigned char LowerToUpper(unsigned char c)
{
        if(c >= 'a' && c <= 'z')
                return (c – 0x20);
        else
                return c;
}
```

图4-7 使用函数原型的程序

176

使用函数原型的一个重要好处是，如果函数原型与实际的函数定义不匹配，mikroC在检测到这种情况后，会修改函数调用中的数据类型，以保持同函数原型中声明的数据类型一致。

假设有如下的一段代码：

```
unsigned char c = 'A';
unsigned int x = 100;
long Tmp;
long MyFunc(long a, long b);        // function prototype

void main()
```



世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

```
{
        ...............
        ...............
        Tmp = MyFunc(c, x);
        ...............
        ...............
}
```

在这个例子里，因为在函数原型里定义了两个long类型的参数，所以在函数MyFunc使用变量c和x之前，它们被转换成long类型。

4.1.2 向函数传递数组

在很多情况下，有可能需要向函数传递数组。传递单个的数组元素是很简单的，只要简单地指定传递数组元素的索引号就可以。例如，在下面的函数调用里，将数组A的第二个元素(索引为1)传递到函数Calc里。相当重要的是，单个的数组元素是通过值来传递的（即将一个数组元素的值复制到函数里）：

```
x = Calc(A[1]);
```

在某些应用里，可能需要向函数传递整个数组。将数组的名称当作参数传递给函数，这样就允许传递整个的数组。要传递整个数组，则必须将数组名包含在括号内。在正式的参数声明中，无需指定数组的大小。在函数头部分，必需指定数组名，并附加一对空的方括号。重要的是，当向函数传递整个数组时，实际上传递的不是数组的副本，而是数组中第一个元素的地址（也就是，全部数组元素是通过引用来传递的，即在函数中可以修改原始的数组元素）。 177

下面的一些例子描述了怎样向函数传递一个完整的数组。

例4.6 编写一段程序，将从1到10的数字存储到数组Number中。然后调用函数Average，计算其平均值。

解 图4-8给出了满足要求的程序清单。函数Average接收数组Number的元素，然后计算数组元素的平均值。

```
/***********************************************************************

              PASSING AN ARRAY TO A FUNCTION
              ==============================

This program stores numbers 1 to 10 in an array called Numbers. Function
Average is then called to calculate the average of these numbers.


Programmer:  Dogan Ibrahim
File:        AVERAGE.C
Date:        May, 2007
***********************************************************************/

/* Function to calculate the average */
float Average(int A[ ])
{
        float Sum = 0.0, k;
        unsigned char j;

        for(j=0; j<10; j++)
        {
                Sum = Sum + A[j];
```

178

图4-8 向函数传递数组的程序

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
        }
        k = Sum / 10.0;
        return k;
}


/* Start of the main program */
void main()
{
        unsigned char j;
        float Avrg;
        int Numbers[10];

        for(j=0; j<10; j++)Numbers[j] = j+1;
        Avrg = Average(Numbers);
}
```

图4-8 （续）

例4.7 重复例4.6，但是这次在程序开始处定义数组的大小，然后将该数组的大小传递给函数。

解 所需的程序清单如图4-9所示。

```
/************************************************************************

                PASSING AN ARRAY TO A FUNCTION
                ================================

This program stores numbers 1 to N in an array called Numbers where N is
defined at the beginning of the program. Function Average is then called to
calculate the average of these numbers.


Programmer:  Dogan Ibrahim
File:        AVERAGE2.C
Date:        May, 2007
*************************************************************************/

#define Array_Size 20

/* Function to calculate the average */
float Average(int A[ ], int N)
{
        float Sum = 0.0, k;
        unsigned char j;

        for(j=0; j<N; j++)
        {
                Sum = Sum + A[j];
        }
        k = Sum / N;
        return k;
}


/* Start of the main program */
void main()
{
        unsigned char j;
        float Avrg;
        int Numbers[Array_Size];

        for(j=0; j<Array_Size; j++)Numbers[j] = j+1;
        Avrg = Average(Numbers, Array_Size);
}
```

图4-9 另一个向函数传递数组的程序

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

也可以使用指针来传递整个数组。将数组的第一个元素的地址传送给函数，函数就可以使用指针按照需要来操作数组。请看下面的例子。

例4.8 重复例4.6，但是这次使用指针来向函数传递数组。

解 图4-10给出了满足要求的程序清单。使用一个整型指针来向函数传递数组元素，并使用指针来操作函数中的数组元素。注意，数组第一个元素的地址作为整型数值传递，如语句`&Numbers[0]`。

```
/*****************************************************************

             PASSING AN ARRAY TO A FUNCTION
             ================================

This program stores numbers 1 to 10 in an array called Numbers. Function
Average is then called to calculate the average of these numbers.


Programmer:  Dogan Ibrahim
File:        AVERAGE3.C
Date:        May, 2007
*****************************************************************/

/* Function to calculate the average */
float Average(int *A)
{
        float Sum = 0.0, k;
        unsigned char j;

        for(j=0; j<10; j++)
        {
                Sum = Sum + *(A + j);
        }
        k = Sum / 10.0;
        return k;
}


/* Start of the main program */
void main()
{
        unsigned char j;
        float Avrg;
        int Numbers[10];

        for(j=0; j<10; j++)Numbers[j] = j+1;
        Avrg = Average(&Numbers[0]);
}
```

图4-10 用指针传递数组的程序

4.1.3 通过引用向函数传递变量

在默认的情况下，函数的参数是通过值来传递的。虽然这种方法有很多特别的优点，但在某些情况下传递参数的地址，也就是通过引用来传递参数，更合适也更有效率。因为在传送变量地址时，原始的参数值可以被函数修改，所以函数不必返回任何变量。在下面的例子里，可以看到如何传递参数的地址，以及参数的值是如何在函数里被修改的。

例4.9 编写一个函数Swap，接收两个整型参数，然后交换这两个参数的值。在主程序里

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

调用此函数来交换两个变量的值。

解 图4-11给出了满足要求的程序清单。因为函数Swap并没有返回任何数据，所以被定义为void类型。这里有两个参数a和b，在函数的开头使用两个整型指针来传递这两个变量的地址。在函数体内，通过在参数前面插入符号“*”可以访问参数的值。在主程序内，通过在变量名前面使用符号“&”来向函数传递变量的地址。在程序的最后，变量p和q分别被设置为20和10。 [180]

```
/*****************************************************************

                PASSING VARIABLES BY REFERENCE
                ==============================

This program shows how the address of variables can be passed to functions.
The function in this program swaps the values of two integer variables.


Programmer:  Dogan Ibrahim
File:        SWAP.C
Date:        May, 2007
*****************************************************************/

/* Function to swap two integers */
void Swap(int *a, int *b)
{
        int temp;

        temp = *a;                  // Store a in temp
        *a = *b;                    // Copy b to a
        *b = temp;                  // Copy temp to b
}


/* Start of the main program */
void main()
{
        int p, q;

        p = 10;                     // Set p = 10
        q = 20;                     // Set q = 20
        swap(&p, &q);               // Swap p and q (p=20, q=10)
}
```

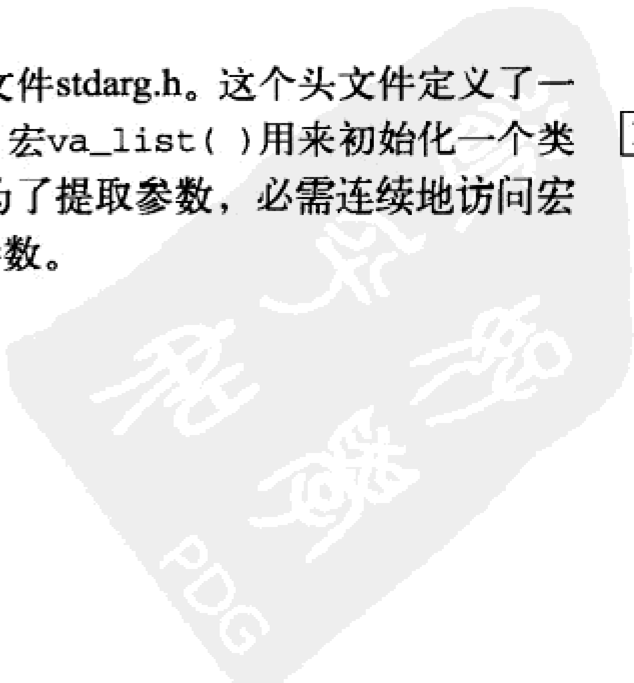
图4-11 通过引用向函数传递变量

4.1.4 参数数量可变

省略号（“...”）是由3个连续的而没有空格隔开的句点组成。省略号在函数原型中可用来指明参数的可变数量或者具有不同数据类型的参数。在下面的例子里，使用省略号声明了一个函数原型。在这个声明里，当函数被调用时，必需提供至少两个整型的参数。另外，还可以提供任意多的附加参数。 [181]

```
unsigned char MyFunc (int a, int b,...);
```

要使用可变数量的参数，则必需在程序的开始处包含头文件stdarg.h。这个头文件定义了一个新的数据类型va_list，它实际上是一个字符指针。另外，宏va_list( )用来初始化一个类型为va_list的对象为指向函数中第一个附加参数的地址。为了提取参数，必需连续地访问宏va_arg( )，并使用字符指针和参数类型作为va_arg( )的参数。 [182]



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

图4-12给出了程序实例。在程序中，函数开始处只声明了一个int类型的参数，省略号用来声明可变数量的参数。变量num是调用程序传递的参数数目。使用宏va_arg(ap,int)读入参数，然后用变量temp将其相加，最后从函数返回结果。

183

```
/*******************************************************************

         PASSING VARIABLE NUMBER OF ARGUMENTS
         ====================================

This program shows how variable number of arguments can be passed to a
function. The function header declares one integer variable and an ellipsis is
used to declare variable number of parameters. The function adds all the
arguments and returns the sum as an integer. The number of arguments is
supplied by the calling program as the first argument to the function.

Programmer:  Dogan Ibrahim
File:        VARIABLE.C
Date:        May, 2007
*******************************************************************/

#include <stdarg.h>

/* Function with variable number of parameters */
int Sum(int num,...)
{
        unsigned char j;
        va_list ap;
        int temp = 0;

        va_start(ap, num);

        for(j = 0; j < num; j++)
        {
                temp = temp + va_arg(ap, int);
        }
        va_end(ap);
        return temp;
}


/* Start of the main program */
void main()
{
        int p;

        p = Sum(2, 3, 5);                // 2 arguments. p=3+5=8
        p = Sum(3, 2, 5, 6);             // 3 arguments, p=2+5+6=13
}
```

图4-12 向函数传递参数变量

4.1.5 函数的可重入性

mikroC编译器只支持有限的函数可重入功能。没有参数和局部变量的函数可以从中断服务子程序和主程序中调用。而带有参数/或局部变量的函数只能从中断服务子程序或主程序中调用。

4.1.6 静态函数变量

一般来说，在主程序的前面，在程序开头处声明的变量都是全局变量，其值是可以在程序其他任何部分访问和修改的。在函数中声明一个全局变量，可以确保在不同的函数调用之间变

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

www.藏书

量的值可以随时更新。但是这样也破坏了变量的保密性，降低了函数在不同应用之间的可移植性。一个更好的办法是，把这些变量声明为static（静态）。静态变量主要在定义函数时使用。当变量被声明为静态后，在不同的函数调用之间，变量的值是可以保留的。在下面的例程里，变量k被声明为静态变量，且被初始化为0。在退出函数之前，这个变量的值将会自增量。变量k将会保持它上一次得到的值，直到下次调用（例如，在第2次调用时，k的值将变为1）：

```
void Cnt(void)
{
      static int k=0;    //声明静态变量k
      .........
      .........
      k++;               //k自增
}
```

4.2 mikroC 的内置函数

mikroC编译器提供了一系列的内置函数以供程序调用。表4-1给出了这些函数及其简要的描述。在这些函数中，大多数并不需要包含头文件就可以在程序中使用。

184

表4-1　mikroC的内置函数

函　数	描　述
Lo	返回一个数的最低字节（0～7位）
Hi	返回一个数的次低字节（8～15位）
Higher	返回一个数的次高字节（16～23位）
Highest	返回一个数的最高字节（24～31位）
Delay_us	设置软件延时，以微秒为单位
Delay_ms	设置定常的软件延时，以毫秒为单位
Vdelay_ms	使用程序变量设置延时，以毫秒为单位
Delay_Cyc	基于微控制器时钟设置延时
Clock_Khz	返回微控制器时钟，以频率（kHz）为单位
Clock_Mhz	返回微控制器时钟，以频率（Mhz）为单位

函数Lo、Hi、Higher和Highest例外，它们需要头文件built_in.h。更详细的说明，请参阅mikroC技术手册。

函数Delay_us和Delay_ms常用于需要延时的程序中（例如，用来闪烁一个LED灯）。下面给出了使用函数Delay_ms的例子。

例4.10　在图4-13中，一个LED通过一个限流电阻连接到PIC18FXXX的PORTB端口的位0（即RB0引脚）。确定限流电阻值，并编程使LED以1秒为周期不停地闪烁。

解　将LED灯连接到微控制器有两种模式：电流灌入和电流拉出。在当前的电流灌入模式（如图4-14所示）下，LED灯的一端接+5 V电源，而另一端则通过限流电阻连接到微控制器的输出端口引脚。

185

在正常的工作条件下，LED两端的电压约为2 V，流过LED的电流约为10 mA（一些低功耗的LED灯可在低至1 mA的电流下工作）。PIC控制器输出端口能拉出或者灌入的最大电流为25 mA。

可以这样计算限流电阻的适当值：在电流灌入模式下，当控制器输出为逻辑0（即约为0 V）时，点亮LED灯。则所需的电阻值为：

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

$$R = \frac{5V - 2V}{10\ mA} = 0.3\ k\Omega$$

186　所以，可以选择290Ω的电阻（如果要减小电流和降低亮度，也可以选择稍大一点的电阻）。

在电流拉出模式下（如图4-15所示），LED灯的一端连接到微控制器的输出端口，而另一端则通过一个限流电阻接地。当微控制器输出为逻辑1(即约为5 V)时，点亮LED灯。在电流灌入模式和电流拉出模式下可以使用同样的电阻值。

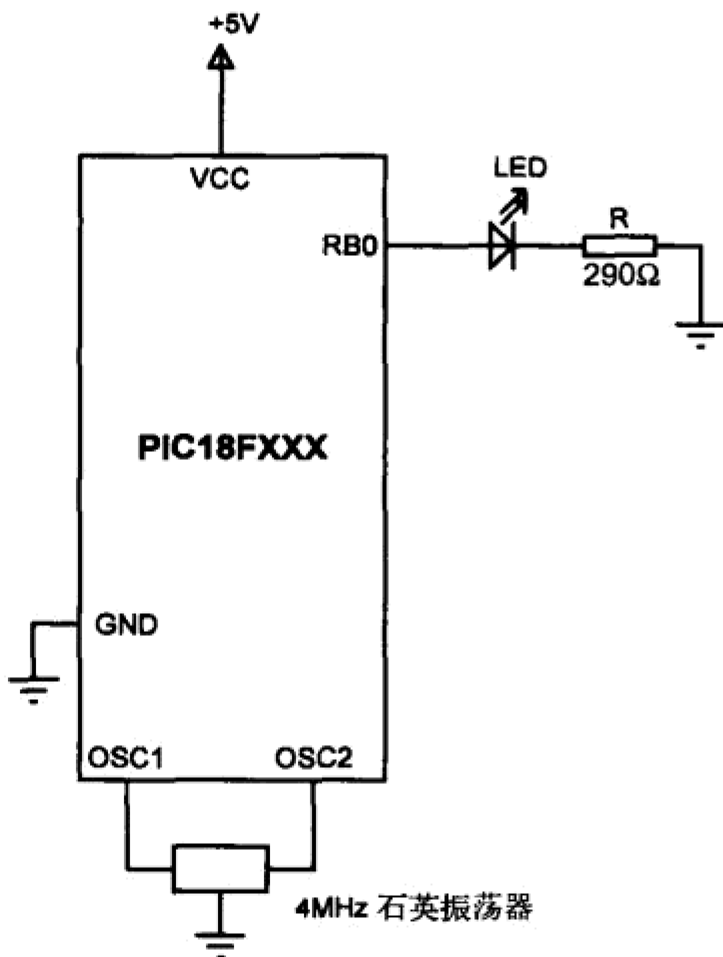


图4-13　连接到PIC微控制器RB0口的LED灯

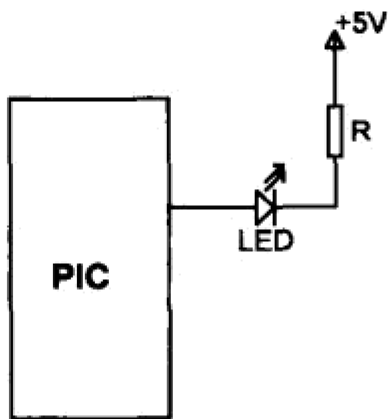


图4-14　电流灌入模式下的LED连接

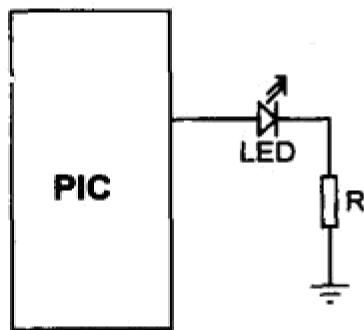


图4-15　电流拉出模式下的LED连接

满足要求的程序清单如图4-16（程序FLASH.C）所示。在程序开头处，使用语句TRISB=0将PORTB端口配置为输出端口。使用for语句创建了一个无限循环，在循环内LED灯以1s的间隔闪烁。

如图4-17（程序FLASH2.C）所示，可以很容易地使用define语句来改进图4-16的程序，使之使用起来更友好。187

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源


```
/*******************************************************************

                        FLASHING AN LED
                        ===============

This program flashes an LED connected to port RB0 of a microcontroller
with one second intervals. mikroC built-in function Delay_ms is used to
create a 1 second delay between the flashes.

Programmer:  Dogan Ibrahim
File:        FLASH.C
Date:        May, 2007
*******************************************************************/

void main()
{
        TRISB = 0;                      // Configure PORTB as output
        for(; ;)                        // Endless loop
        {
                PORTB = 1;              // Turn ON LED
                Delay_ms(1000);         // 1 second delay
                PORTB = 0;              // Turn OFF LED
                Delay_ms(1000);         // 1 second delay
        }
}
```

图4-16 闪烁LED灯的程序

```
/*******************************************************************

                        FLASHING AN LED
                        ===============

This program flashes an LED connected to port RB0 of a microcontroller
with one second intervals. mikroC built-in function Delay_ms is used to
create a 1 second delay between the flashes.

Programmer:  Dogan Ibrahim
File:        FLASH2.C
Date:        May, 2007
*******************************************************************/

#define LED PORTB.0
#define ON  1
#define OFF 0
#define One_Second_Delay Delay_ms(1000)

void main()
{
        TRISB = 0;                      // Configure PORTB as output

        for(; ;)                        // Endless loop
        {
                LED = ON;               // Turn ON LED
                One_Second_Delay;       // 1 second delay
                LED = OFF;              // Turn OFF LED
                One_Second_Delay;       // 1 second delay
        }
}
```

图4-17 闪烁LED灯的另一个程序

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源



4.3 mikroC 的函数库

mikroC编译器拥有大量可用的库函数。这些库函数可以在程序任意部分调用，而且不需要在程序中包含头文件。mikroC用户手册给出了每个库函数的详细描述和例程。本节将认识了解这些有用的库函数，并给出一些重要且常用的库函数的详细解释和例程。

表4-2按功能顺序列出了mikroC的库函数。

表4-2 mikroC库函数

库	描 述	库	描 述
ADC	数模转换函数	PWM	PWM函数
CAN	CAN总线函数	RS-485	RS-485通信函数
CANSPI	基于SPI的CAN总线函数	Sound	音频函数
Compact Flash	紧凑型闪存函数	SPI	SPI总线函数
EEPROM	EEPROM存储器读/写函数	USART	USART串行通信函数
Ethernet	以太网函数	Util	效用函数
SPI Ethernet	基于SPI的以太网函数	SPI Graphics LCD	基于SPI的图形LCD函数
Flash Memory	闪存函数	Port Expander	端口扩展函数
Graphics LCD	标准图形LCD函数	SPI LCD	基于SPI的LCD函数
T6963C Graphics LCD	基于T6963的图形LCD函数	ANSI C Ctype	C语言的字符函数
I^2C	I^2C总线函数	ANSI C Math	C语言的数学函数
Keypad	键盘函数	ANSI C Stdlib	C语言的标准函数
LCD	标准LCD函数	ANSI C String	C语言的字符串函数
Manchester Code	曼彻斯特编码函数	Conversion	转换函数
Multi Media	多媒体函数	Trigonometry	三角函数
One Wire	One Wire函数	Time	时间函数
PS/2	PS/2函数		

以下是一些常用的库函数：

- EEPROM库
- LCD库
- 软件UART库
- 硬件USART库
- 音频库
- ANSI C库
- 混合库

188

4.3.1 EEPROM 库

EEPROM库既包括从片上PIC微控制器的非易失性EEPROM存储器读取数据的函数，又包括向这个存储器写入数据的函数。EEPROM库提供以下两个函数：

189～190

- Eeprom_Read
- Eeprom_Write

函数Eeprom_Read用来从EEPROM中一个指定的地址读取1 B数据。由于地址是整型数据，所以这个函数可以支持大于256 B的PIC微控制器。在从EEPROM连续两次读取数据之间，应该有一个20ms的延时，以保证返回正确的数据。在下面的例子中，读取EEPROM中地址为0x1F的

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

储存单元中的字节数据，并存储在变量Temp里：

```
Temp=Eeprom_Read(0x1F);
```

函数Eeprom_Write用来向EEPROM中一个指定的地址写入1B数据。由于地址是整型数据，所以这个函数可以支持大于256B的PIC微控制器。在向EEPROM连续两次读/写数据之间，应该有一个20ms的延时，以保证EEPROM数据的正确传输。在下面的例子中，将数据0x05写入到EEPROM中地址为0x2F的储存单元中。

```
Eeprom_Write(0x2F,0x05);
```

例4.11 编制一个程序，读取EEPROM中地址从0到0x2F单元的数据，然后将数据传送到PIC微控制器的PORTB端口。

解 图4-18给出了满足要求的程序代码。使用for循环来读入EEPROM中的数据，并传送到PIC微控制器的PORTB端口。注意，在每两次连续读取操作之间都有20 ms的延时。 191

```
/*****************************************************************************

                    READING FROM THE EEPROM
                    ========================

This program reads data from addresses 0 to 0x2F of the EEPROM and then
sends this data to PORTB of the microcontroller.

Programmer:  Dogan Ibrahim
File:        EEPROM.C
Date:        May, 2007
*****************************************************************************/

void main()
{
        unsigned int j;
        unsigned char Temp;

        TRISB = 0;                              // 将PORTB配置为输出

        for(j=0; j <= 0x2F; j++)
        {
                Temp = Eeprom_Read(j);
                PORTB = Temp;
                Delay_ms(20);
        }
}
```

图4-18 读入EEPROM数据的程序

4.3.2 LCD 库

所有微控制器都缺少视频显示功能。视频显示将使得微控制器的应用更加友好，而且能以比七段数码显示管、LED显示和字符显示更多的方式来显示文本信息、图形和数值。标准的视频显示器需要复杂的接口和较高的成本。LCD是字母数字（或者图形）显示器，常用在基于微控制器的应用中。这些显示设备具有不同的形状和大小。一些LCD具有超过40个字符的显示长度，可以显示几行字符。还有些LCD还可以被编程，用来显示图形和图像。有的模块支持彩色显示，而有的模块还集成了背光系统以适应昏暗的环境。

就接口技术而言，目前的LCD主要有两种：并行和串行。并行LCD（例如日立HD44780系列）使用几条线（通常是4条或者8条数据线）连接到微控制器电路。而串行LCD只使用一条数 192

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

据线连接到微控制器，并通过RS232异步数据通信协议来传输数据。一般来说，串行LCD使用起来更简便，但成本也较并行LCD高。本书只讨论并行LCD，因为它们在基于微控制器的工程中更常用。

并行LCD的低级编程往往是一项很复杂的工作，要求程序人员对LCD的内部操作（包括时序图）有深刻的理解。幸而mikroC语言为基于文本的和基于图像的LCD提供了可用的函数，在基于PIC微控制器的应用中简化了LCD的使用。

在基于LCD的微控制器应用中，HD44780控制器是常用的器件。下面简要描述HD44780控制器，并给出一些商业应用的LCD模块信息。

HD4780 LCD控制器

HD44780是一种广泛使用在工业和商业领域的LCD控制器，为广大爱好者所青睐。该模块支持黑白显示，并有不同的形状和大小，可用的模块有8、16、20、24、32或40个字符的不同显示规格。根据不同的模块，显示器提供14脚或16脚的接口。表4-3给出了常见的14脚LCD的引脚配置和引脚功能描述。

Vss（第1引脚）。是0V电压或地。

V_{DD}（第2引脚）。该引脚应接到正电源。尽管使用手册指定使用5V直流供电，但该模块通常允许工作在低至3V或高至6V的环境下。

V_{EE}（第3引脚）。是对比度控制引脚。它应该连接到直流正电源，用来调节显示的对比度。常常把滑动变阻器的电刷跟这个引脚连接，而变阻器的另一根引线则接地。这样就可以根据需要很方便地调节V_{EE}的电压值，进而调节显示对比度。

RS（第4引脚）。是寄存器选择引脚。当引脚为低电平时，传输到LCD的数据被当作控制命令。当引脚为高电平时，可以将字符数据从LCD中读出或者向LCD写入。

193

R/W（第5引脚）。读写引脚。为了向LCD模块写入数据，需要将此引脚置为低电平。当此引脚被置为高电平时，则可以从LCD读出字符数据或状态信息。

表4-3　HD44780 LCD模块的引脚配置

引脚编号	名　称	功　能	引脚编号	名　称	功　能
1	V_{SS}	接地	8	D1	数据位1
2	V_{DD}	接+ve电源	9	D2	数据位2
3	V_{EE}	对比度电压	10	D3	数据位3
4	RS	寄存器选择	11	D4	数据位4
5	R/W	读／写	12	D5	数据位5
6	EN	使能	13	D6	数据位6
7	D0	数据位0	14	D7	数据位7

EN（第6引脚）。使能端引脚。用来在模块与微控制器之间发起命令或者数据的传输。当向显示器写入时，数据的传输仅发生在该引脚从高电平向低电平跳变期间。而从显示器进行读取时，数据在EN脚电平从低电平跳变为高电平后才可以被读出，在高电平期间数据都保持可用。

D0～D7（第7～14引脚）。八位数据总线（D0～D7）。数据可以8位或者两个4位的半字节形式在微控制器和LCD模块之间传输。对于后一种情况的应用，只有高4位（D0～D3）被使用。4位传输的优点在于，可以用较少的I/O线同LCD进行通信。

194

mikroC的LCD库提供了大量函数，使用4位和8位数据接口来控制基于文本的LCD和图像LCD。最常见的是4位接口的基于文本的LCD。本节将描述这些LCD可用的mikroC函数。关于其他的基于文本或者图像的LCD的函数，更多详情请参阅mikroC技术手册。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

下面是4位接口的基于文本的LCD函数：

- Lcd_Config
- Lcd_Init
- Lcd_Out
- Lcd_Out_Cp
- Lcd_Chr
- Lcd_Chr_Cp
- Lcd_Cmd

Lcd_Config 函数Lcd_Config用来配置LCD的接口。LCD和微控制器之间默认的连接如下：

LCD	微控制器端口引脚
RS	2
EN	3
D4	4
D5	5
D6	6
D7	7

LCD的R/W引脚没有被使用，应该接地。

该函数可以用来改变默认的连接。在函数调用时，应该按下面的顺序使用参数：

```
port name, RS pin, EN pin, R/W pin, D7 pin, D6 pin, D5 pin, D4 pin
```

端口名字应该指定为它们的地址。例如，若RS脚连接到RB0，EN脚连接到RB1，D7脚连接到RB2，D6连接到RB3，D5脚连接到RB4，D4脚连接到RB5，那么这个函数的调用如下： 195

```
Lcd_Config (&PORTB, 0, 1, 2, 3, 4, 5);
```

Lcd_Init 如上述一样配置好默认连接后，函数Lcd_Init用来配置微控制器和LCD之间的接口。端口名字应该指定为它们的地址。例如，假设LED连接到PORTB，并使用了默认的连接，则函数的调用如下：

```
Lcd_Init (&PORTB);
```

Lcd_Out 函数Lcd_Out用来在LCD上的指定行和列显示文本。在函数调用时，需要用到的参数应按顺序排列如下：

```
row, column, text
```

例如，要在第1行、第2列显示文本“Computer”，则函数调用如下：

```
Lcd_Out (1, 2, "Computer");
```

Lcd_Out_Cp 函数Lcd_Out_Cp用来在当前光标位置显示文本内容。例如，要在当前指针位置显示的内容是“Computer”，则函数应该这样调用：

```
Lcd_Out_Cp ("Computer");
```

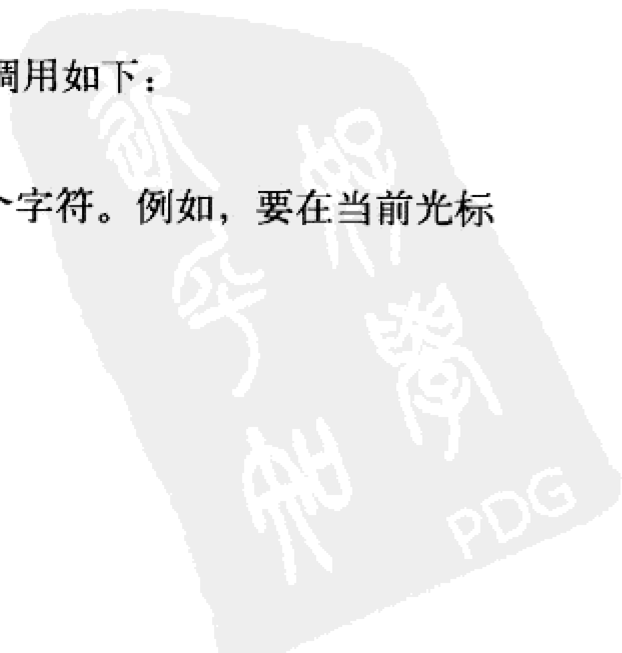
Lcd_Chr 函数Lcd_Chr用来在指定行和列的位置显示一个字符。函数调用所需参数的顺序排列如下：

```
row, column, character
```

例如，要在LCD第2行、第4列的位置显示字符“K”，则函数调用如下：

```
Lcd_Chr (2, 4, 'K');
```

Lcd_Chr_Cp 函数Lcd_Chr_Cp用来在当前光标位置显示一个字符。例如，要在当前光标位置显示字符“M”，则函数调用如下：



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
Lcd_Chr_Cp ('M');
```

196

Lcd_Cmd　函数Lcd_Cmd用于向LCD发送命令。使用命令，可以实现将光标移动到任意行、清空LCD屏幕、闪亮光标或者移位显示等功能。表4-4给出了最常用的LCD命令。例如，要清空LCD屏幕，则函数调用如下：

```
Lcd_Cmd (Lcd_Clear);
```

表4-4　LCD命令

LCD命令	描　述
LCD_CLEAR	清空LCD屏幕
LCD_RETRUN_HOME	返回光标到开始位置
LCD_FIRST_ROW	移动光标到第1行
LCD_SECOND_ROW	移动光标到第2行
LCD_THIRD_ROW	移动光标到第3行
LCD_FOURTH_ROW	移动光标到第4行
LCD_BLINK_CURSOR_ON	闪亮光标
LCD_TURN_ON	打开显示
LCD_TURN_OFF	关闭显示
LCD_MOVE_CURSOR_LEFT	左移光标
LCD_MOVE_CURSOR_RIGHT	右移光标
LCD_SHIFT_LEFT	向左移位显示
LCD_SHIFT_RIGHT	向右移位显示

下面的例子说明了LCD的初始化和使用。

例4.12　如图4-19所示，一个基于文本的LCD以默认模式连接到PIC18F452微控制器上。编写一个程序，向LCD的第1行、第4列输出文本“My Computer”。

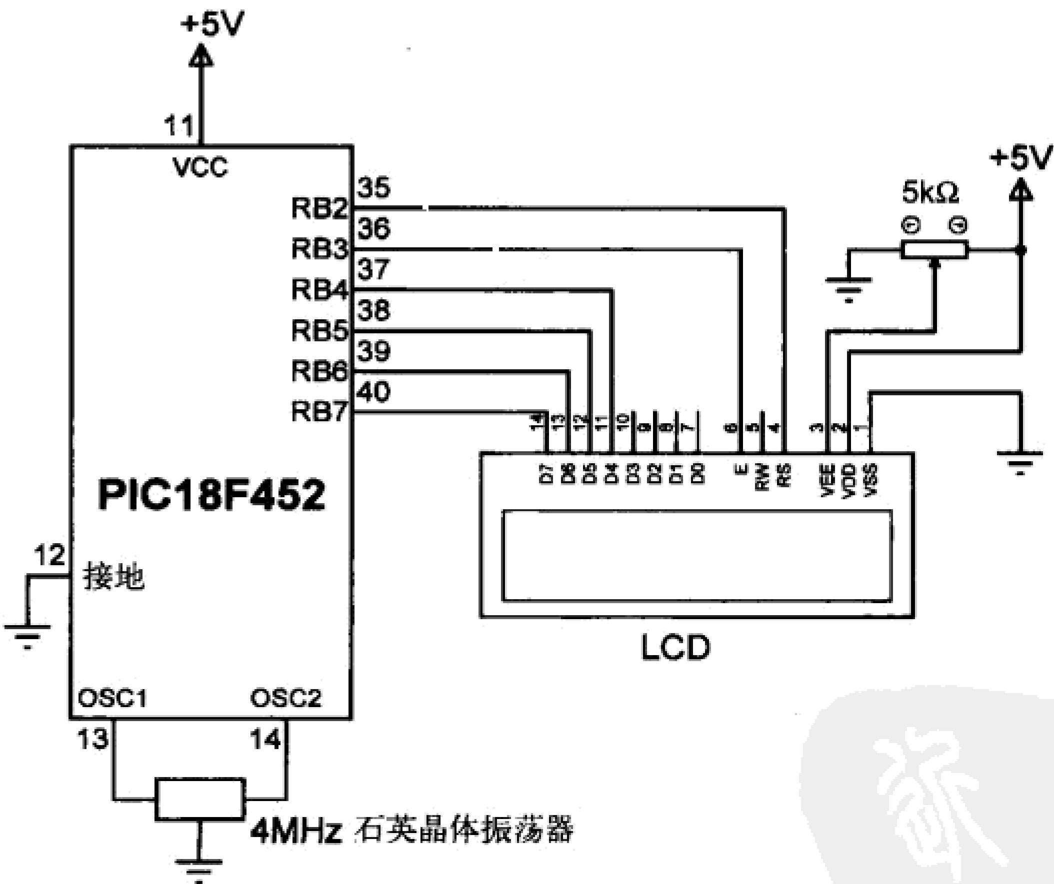


图4-19　LCD和PIC微控制器的连接

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

197
~
198

解 满足要求的程序清单如图4-20（程序LCD.C）所示。在程序开头，使用语句TRISB=0将PORTB配置为输出端口。然后初始化LCD，清空LCD屏幕，再显示出文本信息“My Computer”。

```
/*********************************************************************
              WRITING TEXT TO AN LCD
              ======================

A text based LCD is connected to a PIC microcontroller in the default mode.
This program displays the text "My Computer" on the LCD.

Programmer:  Dogan Ibrahim
File:        LCD.C
Date:        May, 2007
*********************************************************************/

void main()
{
        TRISB = 0;                         // Configure PORTB as output

        Lcd_Init(&PORTB);                  // Initialize the LCD
        Lcd_Cmd(LCD_CLEAR);                // Clear the LCD
        Lcd_Out(1, 4, "My Computer);       // Display text on LCD
}
```

图4-20 LCD程序清单

4.3.3 软件 UART 库

通用异步收发器（UART）软件库，用于电子设备之间基于RS232的串行通信。在串行通信中，只需要两条电缆（加上一条地线）就可以进行双向传输数据。数据以串行格式逐位地进行传输。一般来说，当传输线引脚（TX）为逻辑1时，接收设备处于空闲状态时，即状态MARK。当这些引脚变为逻辑0时，数据开始传输，即状态SPACE。传输的第一个位是起始位，为逻辑0。接着再传输7个或8个位，然后紧跟着一个奇偶校验位。传输的最后一位是停止位，为逻辑1。串行数据通常以10位的帧形式传输数据，包括1个起始位、8个数据位、1个停止位和不带奇偶校验位。图4-21给出了字符“A”的串行传输过程。字符“A”的ASCII码二进制形式为01000001。如图所示，第一位是起始位，然后是8个数据位，最后是1个停止位。

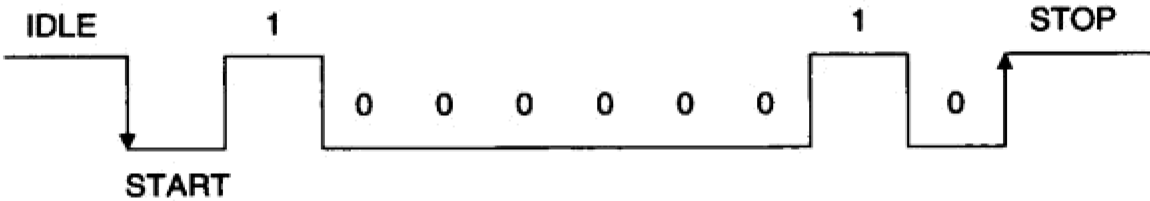


图4-21 串行地传输字符“A”

在串行通信中，位时序是非常重要的，发送（TX）和接收（RX）设备必需有相同的位时序。位时序使用“波特率”来度量，它指定了每秒发送和接收的位数。典型的波特率为4 800、9 600、19 200、38 400，等等。例如，若以9 600的波特率、10位的数据帧格式进行操作，则每秒可以发送或接收960个字符。位与位之间的时间间隔为104 ms。

在基于RS232的串行通信中，两个设备通过一个25针或9针的连接器连接在一起（如图4-22所示）。一般来说，只有TX、RX和GND引脚是必需的。这两种连接器所需的引脚如表4-5所示。

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

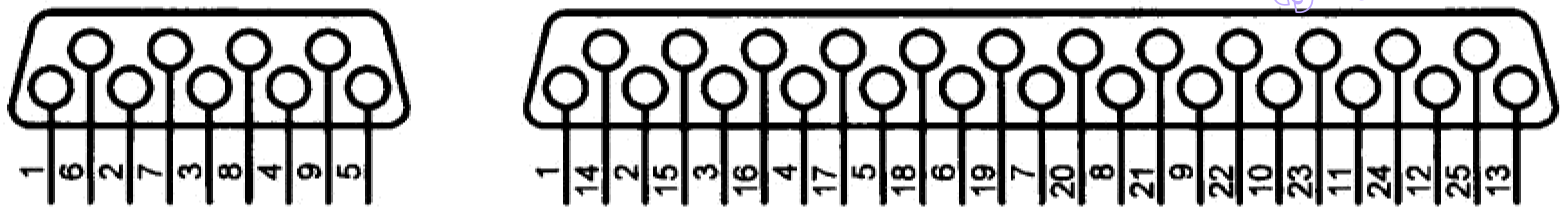


图4-22　25针和9针RS232连接器

表4-5　串行通信所需的针脚

引　　脚	9针连接器	25针连接器
TX	2	2
RX	3	3
GND	5	7

199

RS232通信协议指定的电压值为±12 V。逻辑高电平信号为-12 V，逻辑低电平信号为+12 V。另一方面，PIC微控制器正常工作于0～5 V的电压之间。所以，在输入到微控制器时，必须将RS232信号换成0～5 V。相似地，在向RS232设备发送之前，也必须将微控制器的输出转换成±12V 。人们常用RS232转换芯片来进行电压转换，如Maxim公司生产的MAX232芯片。

串行通信可以硬件方式实现，并使用微控制器的指定引脚，或者用软件来产生所需的信号，并使用微控制器的任意引脚。硬件实现，需要使用一个片上的UART（或者USART）电路，或者连接到微控制器的一个外部UART芯片。基于软件的UART更为常用，而且不需要任何特殊的电路。在基于软件的UART应用中，可使用带延时的循环来产生串行数据。这里只介绍基于软件的UART函数。

mikroC编译器提供以下的软件UART函数：

- Soft_Uart_Init
- Soft_Uart_Read
- Soft_Uart_Write

200

1. Soft_Uart_Init

函数Soft_Uart_Init指定了串行通信所需的参数，并按顺序排列如下：

```
Port, rx pin, tx pin, baud rate, mode
```

port是用作软件UART的端口（如PORTB），rx 是接收引脚号，tx是发送引脚号，baud rate是选择的波特率（其最大值取决于微控制器的时钟频率），mode指定了在端口的输出引脚是否需要将数据取反（“0”表示数据不需要取反，而“1”则表示需要取反）。当使用RS232电压转换芯片的时候，需要将mode设置为0。在创建基于软件的串行通信之前，Soft_Uart_Init必须是被调用的第一个函数。

下面例子的功能是，配置软件UART，将PORTB用作串行口，将RB0用作RX引脚，将RB1用作TX引脚，将波特率设为9 600，且不取反数据：

```
Soft_Uart_Init (PORTB, 0, 1, 9600, 0);
```

2. Soft_Uart_Read

函数Soft_Uart_Read用来从指定的串行端口引脚接收一个字节数据。函数返回的是错误状态，并且从串行端口读入数据。函数并不会等待端口引脚上的数据准备好才进行读入操作，因此必需检测错误参数，以确定是否读入了一个字节数据。正常情况下，错误参数是1，当从串行端口引脚读取一个字节数据后，错误参数变为0。

下面的例子说明了如何调用函数Soft_Uart_Read从配置好的串行端口读入一个字节数

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

据。接收到的数据被存储在变量Temp中：

```
do
    Temp=Soft_Uart_Read (&Rx_Error);
while (Rx_Error);
```

3. Soft_Uart_Write

函数Soft_Uart_Write用来向配置好的串行端口引脚发送一个字节数据。必须将要传送的数据指定为被调用函数的一个参数。 201

例如 将字符“A”发送到串行端口引脚:

```
char MyData='A';
Soft_Uart_Write (MyData);
```

下面的例子说明了软件UART函数的使用。

例4.13 一台PC的串行口（如COM1端口）和一个PIC18F452微控制器相连，PC上的终端仿真软件（如超级终端）用来操作串行口。微控制器的引脚RB0和RB1分别用作引脚RX和TX。要求波特率为9 600。

编制一个程序，从终端读入数据，然后将数据加1，再发送回终端。例如，如果用户输入字符“A”，然后字符“B”将在终端显示出来。假设，使用一个MAX232电压转换芯片将微控制器的信号转换成RS232电平。电路图如图4-23所示。

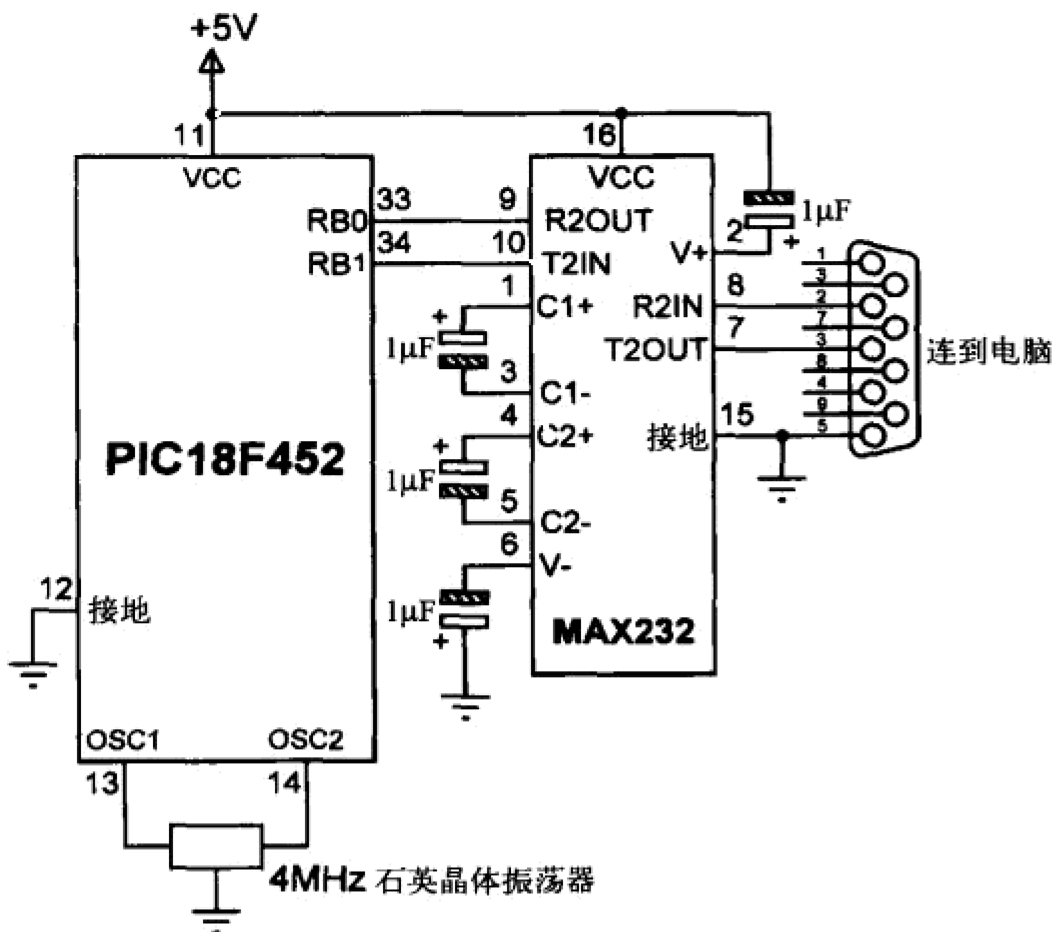


图4-23 例4.13的电路图 202

解 MAX232芯片从微控制器的RB1引脚接收TX信号，然后将其转换成RS232电平。反过来，MAX232芯片接收到的串行数据也将被转换成微控制器适用的电压水平，然后再发送到RB0引脚。注意，正确的MAX232连接需要4个连接到芯片的电容器。

图4-24给出了满足要求的程序清单（程序SERIAL.C）。在程序开头处，调用了函数Soft_Uart_Init来配置串行端口；然后使用for语句形成一个无限循环，接着调用函数Soft_Uart_Read

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

lyw藏书

从终端读入一个字符；最后，将该字节数据加1，并调用函数Soft_Uart_Write将其发送回终端。

```
/*************************************************************************

                READING AND WRITING TO SERIAL PORT
                ==================================

In this program PORTB pins RB0 and RB1 are configured as serial RX and
TX pins respectively. The baud rate is set to 9600. A character is received from a
serial terminal, incremented by one and then sent back to the terminal. Thus, if
character "A" is entered on the keyboard, character "B" will be displayed.

Programmer:  Dogan Ibrahim
File:        SERIAL.C
Date:        May, 2007
**************************************************************************/

void main()
{
        unsigned char MyError, Temp;

        Soft_Uart_Init(PORTB, 0, 1, 9600, 0);        // Configure serial port
        for(; ;)                                      // Endless loop
        {
                do
                {
                        Temp = Soft_Uart_Read(&MyError);     // Read a byte
                } while(MyError);
                Temp++;                                      // Increment byte
                Soft_Uart_Write(Temp);                       // Send the byte
}
```

203

图4-24 例4.13的程序清单

4.3.4 硬件 USART 库

通用同步异步收发器（USART）硬件库包含了大量的函数，它们使用内置于微控制器上的USART电路来发送和接收数据。一些PIC18F系列的微控制器只有一个USART（如PIC18F452微控制器），而另一些则有两个USART电路（如PIC18F520微控制器）。硬件USART比软件实现的USART更有优势，因为它支持更高的波特率，并且在数据被传送到USART的同时微控制器还可以执行其他的操作。

硬件USART库提供了以下函数：

- Usart_Init
- Usart_Data_Ready
- Usart_Read
- Usart_Write

1. Usart_Init

函数Usart_Init用来以指定的波特率对USART进行初始化。这个函数应该是所有的USART函数中最先被调用的。波特率是该函数的唯一参数。在下面的例子中，波特率被设置为9 600：

```
Usart_Init (9600);
```

2. Usart_Data_Ready

函数Usart_Data_Ready用来检测USART是否已经收到一个字节的数据。如果数据已被收到，则函数返回“1”。反之，则返回“0”。此函数不需要参数。下面的程序代码用来检测是否

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

收到了一个字节的数据：

```
if(Usart_Data_Ready())
```

3. **Usart_Read**

函数Usart_Read用来从USART读入一个字节。如果数据没有被接收，则返回“0”。注意，从USART读入数据是一个非阻塞的操作（例如，无论USART是否接收到字节数据，函数都会返回）。函数Usart_Read应该在调用函数Usart_Data_Ready之后才被调用，这样可以确保数据在USART中是可用的。函数Usart_Read没有参数。在下面的例子中，USART将被检测，如果字节数据已经收到，则将数据复制到变量MyData中：

204

```
char MyData;
if (Usart_Data_Read ( )) MyData=Usart_Read( );
```

4. **Usart_Write**

函数Usart_Write用来向USART发送一个字节数据，因此串行数据将从USART发送出去。要传送的数据必需作为函数的一个参数提供。下面的例子中，将字符“A”发送到USART：

```
char Temp='A';
Usart_Write (Temp);
```

下面的例子说明了如何在程序中使用USART硬件函数。

例4.14 一台PC的串行端口（如COM1端口）和一个PIC18F452微控制器相连，PC上的终端仿真软件（如超级终端）用来操作串行端口。微控制器的硬件USART引脚RC7（USART接收引脚RX）和RC6（USART发送引脚TX）通过一块MAX232类型的RS232电压转换芯片连接到PC。波特率被设为9 600。编制一个程序，从终端读入一个数据，然后将其加1，再发送回终端。例如，若用户输入字符“A”，那么字符“B”将在终端上显示出来。例4.14的电路图如图4-25所示。

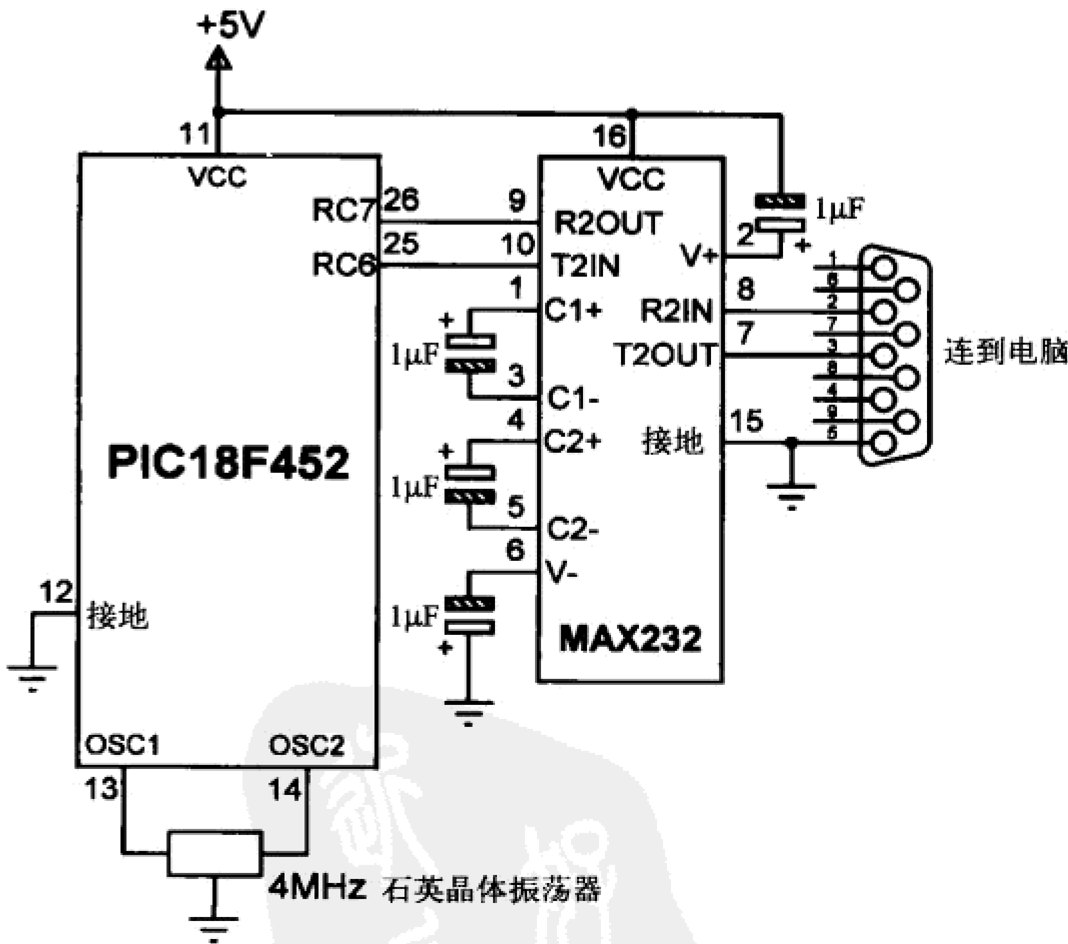


图4-25 例4.14的电路图

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

解 满足要求的程序清单如图4-26所示（程序SERIAL.2）。在程序的开头，调用函数Usart_Init将波特率设为9 600。然后使用for语句来形成一个无限循环。接着调用函数Usart_Data_Ready来检测字符是否准备好，并调用函数Usart_Read来读入字符。在读入字符后，数据被加1，最后通过调用函数Usart_Write将其发送回终端。

```
/*****************************************************************************

      READING AND WRITING TO SERIAL PORT VIA USART
      ==============================================

In this program a PIC18F452 microcontroller is used and USART I/O pins are
connected to a terminal through a MAX232 voltage converter chip. The baud rate is
set to 9600. A character is received from a serial terminal, incremented by one and
then sent back to the terminal. Thus, if character "A" is entered on the keyboard,
character "B" will be displayed.

Programmer:  Dogan Ibrahim
File:        SERIAL2.C
Date:        May, 2007
*****************************************************************************/

void main()
{
        unsigned char MyError, Temp;

        Usart_Init(9600);                          // Set baud rate
        for(; ;)                                   // Endless loop
        {
                while (!User_Data_Ready());        // Wait for data byte
                Temp = Usart_Read();               // Read data byte
                Temp++;                            // Increment data byte
                Usart_Write(Temp);                 // Send the byte byte
        }
}
```

图4-26 例4.14的程序清单

在那些具有两个USART的PIC微控制器中，访问第2个USART时需要在函数尾部添加“2”（例如，Usart_Write2, Usart_Read2，等等）。 205

4.3.5 音频库

音频库里的函数可用来在微控制器应用中产生声音。应该将扬声器（例如压电式扬声器）连接到要求的微控制器端口上。

音频库提供了如下的两个函数：

- Sound_Init
- Sound_Play

1. **Sound_Init**

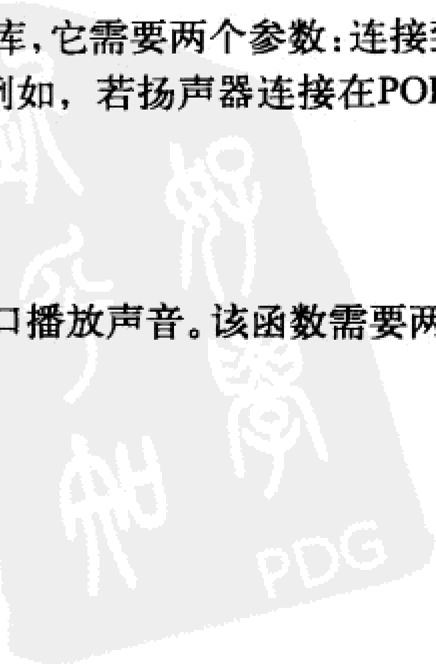
函数Sound_Init用来初始化音频库，它需要两个参数：连接到扬声器的引脚名称和引脚号。需要将端口名称的地址传送给函数。例如，若扬声器连接在PORTB端口的位3上，则函数可作如下的调用：

206

```
Sound_Init (&PORTB, 3);
```

2. **Sound_Play**

函数Sound_Play用来在指定的端口播放声音。该函数需要两个参数：除以10的周期（*TDIV*）



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

和周期数（N）。第一个参数是微控制器循环中将周期除以10。第二个参数是声音持续的时间（时钟周期数）。

第一个参数可用下面的公式来计算：

$$TDIV = \frac{f}{40F}$$

其中，

$TDIV$ 被用作第一个参数的值

F 是要求的声音频率(Hz)

f 是微控制器的时钟频率(Hz)

207

例4.15 编制一个程序，产生1 kHz的声音，假设时钟频率是4 MHz。假设声音持续250个周期。

解 第一个参数的计算如下：

$$TDIV = \frac{f}{40F} = \frac{4 \times 10^6}{40 \times 10^3} = 100$$

由于要求的持续时间是250个周期，所以函数调用如下：

```
Sound_Play (100, 250);
```

4.3.6 ANSI C 库

ANSI C库包括下面的函数（关于这些函数的更多详情，请参阅mikroC用户手册）：

- Ctype library
- Math library
- Stdlib library
- String library

1. Ctype库

Ctype库中的函数主要用于测试和数据转换。表4-6列出了Ctype库中常用的函数。

表4-6 常用的Ctype库函数

函　数	描　述
isalnum	若指定的字符是字母或数字（a~z，A~Z，0~9），则返回“1”
isalpha	若指定的字符是字母（a~z，A~Z），则返回“1”
isntrl	若指定的字符是控制字符（十进制0~31，127），则返回“1”
isdigit	若指定的字符是数字（0~9），则返回“1”
islower	若指定的字符是小写字母，则返回“1”
isprint	若指定的字符是可打印的（十进制数32~126），则返回“1”
isupper	若指定的字符是大写字母，则返回“1”
toupper	将一个字符转换成大写
tolower	将一个字符转换成小写

2. Math库

208

Math库中的函数用于浮点数学运算。表4-7列出了Math库中常用的函数。

3. Stdlib库

Stdlib库包含了标准的库函数，表4-8列出了Stdlib库中常用的函数。

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

tyw藏书

表4-7 常用的Math库函数

函数	描述
acos	返回参数的反余弦值（以弧度为单位）
asin	返回参数的反正弦值（以弧度为单位）
atan	返回参数的反正切值（以弧度为单位）
atan2	返回参数的反正切值（以弧度为单位），其中参数的符号用来确定结果所在的象限
cos	返回参数的余弦值
cosh	返回参数的双曲余弦值
exp	返回参数的指数值
fabs	返回参数的绝对值
log	返回参数的自然对数值
log10	返回参数的以10为底的对数值
pow	返回参数的幂值
sin	返回参数的正弦值
sinh	返回参数的双曲正弦值
sqrt	返回参数的平方根值
tan	返回参数的正切值
tanh	返回参数的双曲正切值

表4-8 常用的Stdlib库函数

函数	描述
abs	返回绝对值
atof	把ASCII字符转换成浮点数
atoi	把ASCII字符转换成整型数
atol	把ASCII字符转换成长整型数
max	返回两数中较大的数
min	返回两数中较小的数
rand	返回0～32 767之间的一个随机数，必须调用函数srand来获得不同序列的数
srand	为函数rand生成一个种子，以产生新序列的数
xtoi	把输入的十六进制字符串转化成整型数

例4.16 编制一个程序，计算0°～90°的三角正弦值。计算步长为1°，将计算结果存放到数组Trig_Sine中。

解 满足要求的程序清单如图4-27所示（程序SINE.C）。使用for语句创建一个循环，并在循环内计算角度的正弦值，然后将值存放到数组Trig_Sine中。注意，在使用函数sin之前，需要将角度转换成弧度值。

209

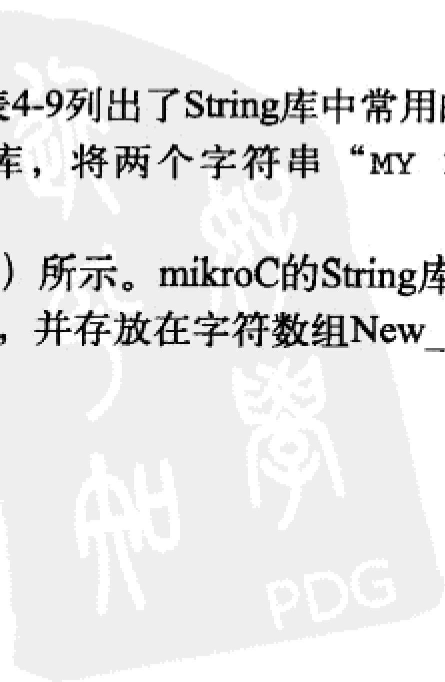
4. String库

String库中的函数用来处理字符串和内存操作。表4-9列出了String库中常用的函数。

例4.17 编制一个程序，说明如何使用String库，将两个字符串“MY POWERFUL”和“COMPUTER”组合成一个新的字符串。

210

解 满足要求的程序清单如图4-28（程序JOIN.C）所示。mikroC的String库函数strcat用来将指针p1和p2指向的两个字符串组成了新的字符串，并存放在字符数组New_String中。



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
/*******************************************************************

        TRIGONOMETRIC SINE OF ANGLES 0 to 90 DEGREES
        ==========================================

This program calculates the trigonometric sine of angles from 0 degrees to
90 degrees in steps of 1 degree. The results are stored in an array called
Trig_Sine.

Programmer:  Dogan Ibrahim
File:        SINE.C
Date:        May, 2007
********************************************************************/

void main()
{
        unsigned char j;
        double PI = 3.14159, rads;

        for(j = 0; j <= 90; j++)
        {
                rads = j * PI /180.0;
                angle = sin(rad);
                Trig_Sine[j] = angle;
        }
}
```

图4-27 计算0° ~90° 的三角正弦值

表4-9 常用的String库函数

函　数	描　述
strcat, strncat	连接两个字符串
strchr, strpbrk	确定字符串中某字符第一次出现的位置
strcmp, strncmp	比较两个字符串
strcpy, strncpy	复制一个字符串到另一个字符串中
strlen	返回字符串的长度

```
/*******************************************************************

                    JOINING TWO STRINGS
                    ===================

This program shows how two strings can be joined to obtain a new string.
mikroC library function strcat is used to join the two strings pointed to by
p1 and p2 into a new string stored in character array New_String.

Programmer:  Dogan Ibrahim
File:        JOIN.C
Date:        May, 2007
********************************************************************/

void main()
{
        const char *p1 = "MY POWERFUL ";        // First string
        const char *p2 = "COMPUTER";            // Second string
        char New_String[80];

        strcat(strcat(New_String, p1), p2);     // join the two strings
}
```

图4-28 使用函数strcat连接两个字符串

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

4.3.7　混合库

211
~
212

混合库中的函数包含了一些进行数据类型转换和执行三角函数计算的子程序。表4-10列出了混合库中常用的函数。

下面的例子说明如何在mikroC语言中应用库子程序。

表4-10　常用的混合库函数

函　　数	描　　述
ByteToStr	将字节数据转换成字符串
ShortToStr	将短整型数据转换成字符串
WordToStr	将无符号数转换成字符串
IntToStr	将整型数据转换成字符串
LongToStr	将长整型数据转换成字符串
FloatToStr	将浮点数转换成字符串
Bcd2Dec	将BCD码转换成十进制数
Dec2Bcd	将十进制数转换成BCD码

例4.18　编制一个程序，根据传递给函数的模式参数值，将由指针p指向的字符串转换成小写或者大写。如果模式参数是非零值，则转换成小写；反之，则转换成大写。要求函数返回指向转换后的字符串的指针。

解　满足要求的程序清单如图4-29所示（程序CASE.C）。程序将检测模式参数的值，如果参数是非零的，则调用函数ToLower把字符串转换成小写；反之，则调用函数ToUpper把字符串转换成大写。最后，程序返回指向转换所得字符串的指针。

```
/************************************************************************

        CONVERT A STRING TO LOWER/UPPERCASE
        ===================================

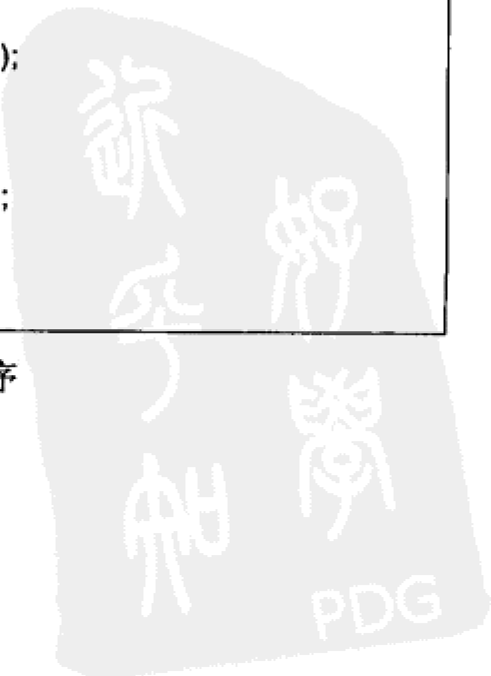
This program receives a string pointer and a mode parameter. If the mode is 1
Then the string is converted to lowercase, otherwise the string is converted to
uppercase.

Programmer:  Dogan Ibrahim
File:        CASE.C
Date:        May, 2007
************************************************************************/

unsigned char *Str_Convert(unsigned char *p, unsigned char mode)
{
        unsigned char *ptr = p;

        if (mode != 0)
        {
                while(*p != '\0')  *p++ = ToLower(*p);
        }
        else
        {
                while(*p != '\0') *p++ = ToUpper(*p);
        }
        return ptr;
}
```

图4-29　例4.18的程序



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

例4.19 编制一个程序，定义一个复数结构体，然后编写将两个复数相加和相减的函数。请说明如何在主程序中调用这些函数。

解 图4-30给出了满足要求的程序清单（程序COMPLEX.C）。在程序开头，创建了一个complex（复数）数据类型的结构体，包括一个实部和一个虚部。函数Add定义了两个复数的加法，并返回复数形式的总和。相似地，函数Subtract定义了两个复数的减法，并返回复数形式的结果。主程序使用了两个复数a和b，即

213

$$a=2.0-3.0j$$
$$b=2.5+2.0j$$

另外还声明了两个其他的复数c和d，复数运算的操作如下：

$$c=a+b, \quad d=a-b$$

```
/*********************************************************************

        COMPLEX NUMBER ADDITION AND SUBTRACTION
        ==========================================

This program creates a data structure called complex having a real part and
an imaginary part. Then, functions are defined to add or subtract two complex
numbers and store the result in another complex number.

The first complex number is,     a = 2.0 – 2.0j
The second complex number is, b = 2.5 + 2.0j

The program calculates, c = a + b
and,                           d = a – b

Programmer:   Dogan Ibrahim
File:              COMPLEX.C
Date:             May, 2007
*********************************************************************/

/* Define a new data type called complex */
typedef struct
{
        float real;
        float imag;
} complex;

/* Define a function to add two complex numbers and return the result as
   a complex number */
complex Add(complex i, complex j)
{
        complex z;

        z.real = i.real + j.real;
        z.imag = i.imag + j.imag

        return z;
}

/* Define a function to subtract two complex numbers and return the result as
   a complex number */
complex Subtract(complex i, complex j)
{
        complex z;
```

图4-30 例4.19的程序

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
        z.real = i.real – j.real;
        z.imag = i.imag – j.imag;

        return z;
}


/* Main program */
void main()
{
        complex a,b,c, d;

        a.real = 2.0;  a.imag =–3.0;             // First complex number
        b.real = 2.5;  b.imag = 2.0;             // second complex number

        c = Add(a, b);                           // Add numbers
        d = Subtract(a, b);                      // Subtract numbers
}
```

图4-30 （续）

214~215

例4.20 一颗炮弹以θ°的角度发射，初始速度为v m/s。d表示炮弹飞行的距离，t表示飞行时间，h表示飞行能达到的最大高度。它们的计算公式如下：

$$h=\frac{v^2\sin\theta}{g}\quad t=\frac{2v\sin\theta}{g}\quad d=\frac{v^2\sin 2\theta}{g}$$

编制一个程序，计算飞行高度、飞行时间和飞行距离。假设，$g=9.81\ \text{m/s}^2$，$v=12$ m/s和$\theta=45^\circ$，调用函数来计算这3个变量。图4-31给出了炮弹的飞行轨迹。

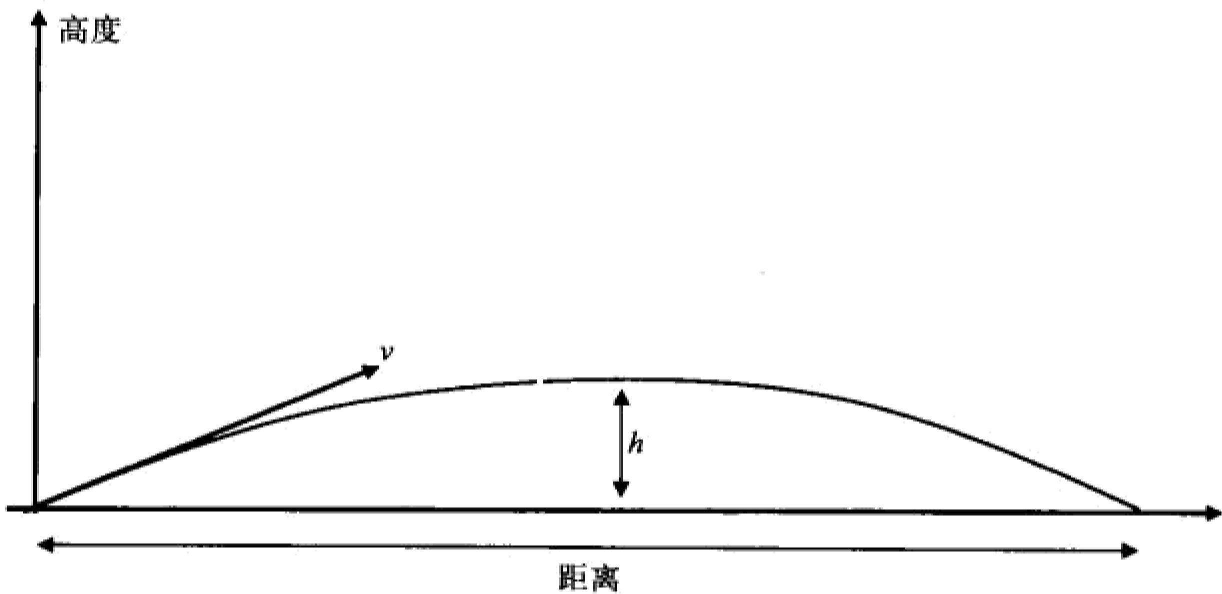


图4-31 炮弹飞行轨迹

216

解 满足要求的程序如图4-32所示（程序PROJECTILE.C）。首先定义了3个函数：函数Height用来计算炮弹飞行的最大高度，函数Flight_time用来计算飞行时间，而函数Distance用来计算飞行距离。另外，函数Radians用来将角度转换成三角函数sine需要的弧度值。在计算出高度、距离和飞行时间后，将它们分别存放在浮点型变量h、d和t中。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
/**************************************************************************

                  PROJECTILE CALCULATION
                  ========================

This program calculates the maximum height, distance traveled, and the flight
time of a projectile. Theta is the firing angle, and v is the initial velocity of the
projectile respectively.

Programmer:  Dogan Ibrahim
File:        PROJECTILE.C
Date:        May, 2007
**************************************************************************/

#define gravity 9.81

/* This function converts degrees to radians */
float Radians(float y)
{
        float rad;

        rad = y * 3.14159 / 180.0;

        return rad;
}


/* Flight time of the projectile */
float Flight_time(float theta, float v)
{
        float t, rad;

        rad = Radians(theta);
        t = (2.0*v*sin(rad)) / gravity;

        return t;
}

float Height(float theta, float v)
{
        float h, rad;

        rad = Radians(theta);
        h = (v*v*sin(rad)) / gravity;

        return h;
}

float Distance(float theta, float v)
{
        float d, rad;

        rad = radians(theta);
        d = (v*v*sin(2*rad)) / gravity;

        return d;
}

/* Main program */
void main()
{
        float theta, v, h, d, t;
```

图4-32　例4.20的程序

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
            theta = 45.0;
            v = 12.0;
            h = Height(theta, v);
            d = Distance(theta, v);
            t = Flight_time(theta, v);
}
```

图4-32 (续)

4.4 小结

本章着重讨论了函数和库。当需要在程序的不同地方重复同一段代码时，调用函数是非常有用的。函数的使用，使得程序更具可读性、更容易管理和维护。一个大的程序可以分成多个函数来独立测试，如果它们都可以正常工作，那么就可以组合起来形成最终的大程序。

217~218

本章还简要地讨论了mikroC的库函数，并给出了如何在主程序中使用这些函数的例子。库函数提供了大量测试成功的可在主程序中调用的子程序，从而简化了程序员的工作。

4.5 练习题

1. 编写一个函数，计算矩形的周长。函数将矩形的两条边以浮点型参数接收，将矩形的周长以浮点数返回。
2. 编写一个主程序，使用练习题1中编写的函数。计算出一个宽为2.3 cm、长为5.6 cm的矩形的周长。然后把结果存放到浮点数MyResult中。
3. 编写一个函数，把英寸转换成厘米。函数使用一个浮点型的英寸值作为参数，然后计算出等效的厘米值。
4. 编写一个主程序，使用练习题3中编写的函数。把12.5英寸转换成厘米值，然后将结果以浮点数类型存放。
5. 一个LED通过一个限流电阻以电流灌入模式连接到PIC18F452微控制器的端口引脚RB0上，编写一个程序，让LED每隔5 s闪烁一次。
6. 八个LED连接到PIC18F452微控制器的端口PORTB上。编写一个程序，令LED实现二进制加法计数。每两次输出之间应有1 s的延时。
7. 一个LED连接到PIC18F452微控制器的端口引脚RB7上。编写一个程序，让LED闪亮，先点亮5 s，然后熄灭3 s，如此循环。
8. 一个基于文本的LCD以4位数据模式连接到PIC18F452微控制器上。编写一个程序，在LCD上显示0～255的计数值。两次计数的时间间隔为1 s。
9. 同练习题8，一个基于文本的LCD连接到PIC18F452微控制器上。编写一个程序，在LED的第一行显示"Exercise 9"。
10. 重复练习题9，但在第1行、第3列处开始显示信息。
11. 一个两行的基于文本的LCD以4位数据模式连接到PIC18F452微控制器上。编写一个程序，在第一行显示文本"COUNTS:"，然后在第二行以2s的时间间隔显示从1到100的计数过程。

219

12. 编写一个程序，计算0°～45°的三角余弦值，计算步长为1°。然后把计算结果存放到一个浮点型数组。
13. 编写一个函数，根据已知的两条直角边，计算并返回直角三角形的斜边长度。请说明如何在主程序中调用该函数来计算一个两直角边分别为4.0 cm和5.0 cm的直角三角形的斜边长度。
14. 编写一个程序，把PIC18F452微控制器的端口引脚RB2配置为RS232串行输出端口。向该端口发送字符"x"，波特率为4 800。
15. PIC18F452微控制器的端口引脚RB0已配置为RS232串行输出端口。请编写一个程序，发送字符串"SERIAL"，波特率为9 600。

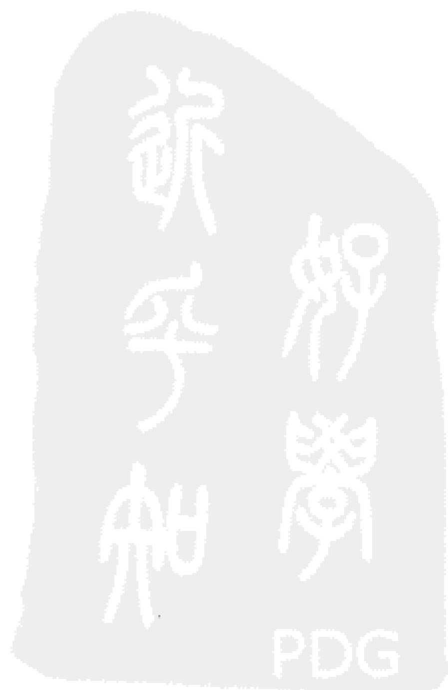
世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

16. 重复练习题15，但使用微控制器芯片上可用的硬件USART。
17. 请说明软件实现的串行数据通信和基于硬件USART的串行通信之间的区别。
18. 编写一个函数，对作为函数参数的两个数组执行加法运算，并把结果存放到其中一个数组中。
19. 编写一个函数，实现关于二维矩阵的如下操作：
    a）矩阵相加，
    b）矩阵相减，
    c）矩阵相乘。
20. 编写一个函数，实现极坐标与直角坐标之间的相互转换。
21. 编写函数，将摄氏温度转换为华氏温度，或者将华氏温度转换为摄氏温度。请说明如何在主程序中调用这些函数，把20℃转化成℉值和把100℉转化成℃值。
22. 编写一个程序，计算函数 $f(x)$ 的值。其中，变量x从0增加到10，步长为0.5，并把计算结果存为数组。假设：

$$f(x)=1.3x^3-2.5x^2+3.1x-4.5$$

220



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

第 5 章　PIC18 开发工具

基于微控制器的系统的开发是一个非常复杂的过程。开发工具包括软件工具和硬件工具，用来帮助程序员以较短的时间开发和测试系统。市面上有各种各样的开发工具，本书不会去全面介绍这方面的知识。本章将简要介绍最常用的开发工具。

对于微控制器系统的软件和硬件开发，主要工具有编辑器、汇编器、编译器、调试器、仿真器、模拟器和设备编程器。在一个典型的开发周期中，首先使用文本编辑器编写应用程序，然后使用汇编器或者编译器把程序翻译成可执行的代码。如果程序包含几个模块，那么使用连接器将它们组合成一个单独的应用程序。任何的语法错误都会被汇编器和编译器检测出来，并且需要在生成可执行的代码前进行改正。接下来，使用仿真器来测试应用程序，无需目标硬件的支持。仿真器在检测那些输入/输出很少甚至没有的算法或者程序的错误时非常有用。仿真可以清除绝大多数的错误。程序成功运行，程序员感到满意后，就使用设备编程器将可执行代码下载到目标微控制器芯片，进行系统级的逻辑测试。诸如内电路调试器和内电路模拟器这样的软件和硬件工具，都可以用来分析程序的操作过程，并可以通过在程序内部设置断点，实时地显示变量和寄存器的状态。

5.1　软件开发工具

软件开发工具是一种通常运行在个人电脑上的计算机程序，它允许程序员（或系统开发人员）创建、修改和测试应用程序。一些常用的软件开发工具有：

- 文本编辑器
- 汇编器/编译器
- 仿真器
- 高级编程语言仿真器
- 集成开发环境（IDE）

5.1.1　文本编辑器

文本编辑器用来创建和编辑程序或文本文件。Windows操作系统自带的文本编辑器叫作记事本。使用记事本，可以创建新的程序文档，修改已存在的文件，显示或打印文件的内容。需要注意的是，如Microsoft Word这样的字处理程序不可以用于此用途，因为它们在文档内嵌有诸如加粗、斜体和下划线之类的文字格式字符。

大多数的汇编器和编译器都带有内置的文本编辑器，因此，从创建程序到汇编或者编译程序，都不必切换编辑器。这些编辑器还提供其他的功能，如关键字自动高亮显示、语法检查、括号匹配和注释行识别。程序的不同部分可以使用不同的颜色显示（例如注释使用一种颜色显示，而关键字则使用另一种颜色显示），这可以增加程序的可读性。这些功能消除了编程阶段中的语法错误，从而加快了开发过程。

PDG

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

lyw藏书

5.1.2 汇编器和编译器

汇编器用来从汇编语言程序生成可执行的代码，然后这些代码才可以下载到基于PIC18的微控制器的内存。编译器则用来从高级编程语言生成可执行的代码。对于PIC18微控制器，最常用的编译器有BASIC、C和PASCAL。汇编语言常用于需要快速处理并且微控制器必需在最短时间内对外部和内部事件做出响应的应用中。然而，使用汇编语言开发一个复杂程序是相当困难的，而且汇编语言程序也难以维护。 222

另一方面，高级编程语言更便于学习，可以在更短时间内开发和测试复杂的程序。高级语言程序也比汇编语言程序更容易维护。

本书只讨论C语言编程。基于PIC18微控制器的程序开发可使用多种C语言编译器。下面是比较流行的几种编译器：

- CCS C（http://www.ccsinfo.com）
- Hi-Tech C（http://htsoft.com）
- C18 C（http://www.microchip.com）
- mikroC C（http://www.mikroe.com）
- Wiz-C C（http://www.fored.co.uk）

尽管大多数C语言编译器在本质上都是一样的，但它们较标准的语言都有不同程度的增加和修改。本书将讨论由mikroElektronika公司开发的C编译器mikroC。

5.1.3 仿真器

仿真器是运行在PC上的计算机程序，无需连接微控制器硬件。通过使用微控制器指令集来解释用户程序指令，可以仿真目标微控制器的操作。随着用户程序的解释执行，仿真器可以显示出寄存器和内存的内容，以及输入/输出端口的状态。还可以在程序里设置断点，在需要的地方检查各个寄存器的内容。另外，用户程序还可以单步方式运行，每按一次按键，程序就执行一条指令，这样就更容易观察寄存器和内存的状态。

一些汇编器带有内置的仿真器。下面是3种常用的内置仿真器的PIC18微控制器汇编器： 223

- MPLAB IDE（http://www.microchip.com）
- Oshon Software PIC18 simulator（http://www.oshonsoft.com）
- Forest Electronics PIC18 assembler（http://www.fored.co.uk）

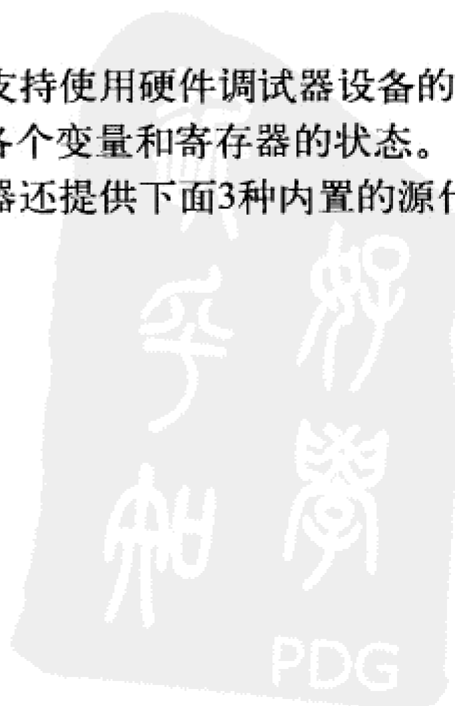
5.1.4 高级编程语言仿真器

高级编程语言仿真器（即源代码级调试器）是运行在PC上的程序，可以检查高级语言程序中的错误。程序员可以在高级语言程序中设置断点，然后观察该断点处程序变量的值、寄存器的内容和存储地址内容。

源代码级调试器还支持使用硬件调试器设备的硬件调试。例如，可以暂停执行目标微控制器上的用户程序，检查各个变量和寄存器的状态。

一些高级语言编译器还提供下面3种内置的源代码级调试器：

- C18 C
- Hi-Tech PIC18 C
- mikroC C



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

5.1.5 集成开发环境（IDE）

集成开发环境（IDE）是功能非常强大的基于PC的程序，包含有编辑、汇编、编译、连接、仿真和源代码级调试程序，然后使用编程设备将生成的可执行代码下载到物理的微控制器芯片。这些程序提供图形界面，用户无需退出程序即可选择各种功能。集成开发环境在开发基于微控制器的系统中特别有用。大部分PIC18高级语言编译器都是IDE，这使得程序员只需要使用一个软件开发工具，就可以完成绝大部分的任务。

5.2 硬件开发工具

PIC18微控制器有很多的硬件开发工具。一些工具是由Microchip公司生产的，而另一些工具是由第三方公司生产的。常见的硬件开发工具有：

224

- 开发板
- 设备编程器
- 内电路调试器
- 内电路模拟器
- 面包板

5.2.1 开发板

开发板是极其有用的微控制器开发工具。简单的开发板只包含一个微控制器和必需的时钟电路。一些复杂的开发板则带有LED、LCD、按钮、串行端口、USB端口、电源电路和设备编程器等。

本节将介绍PIC18微控制器的商用开发板及其特点。

1. LAB-XUSB实验板

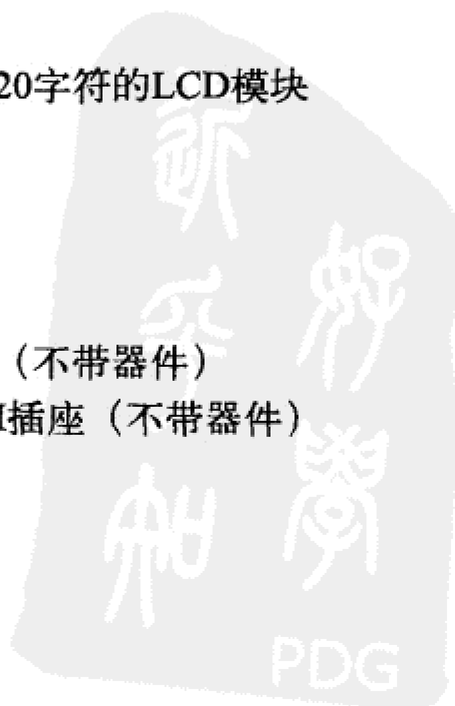
由microEngineering实验室公司生产的LAB-XUSB 实验板（如图5-1所示），常用于40针的基于PIC18微控制器的项目开发中。LAB-XUSB 实验板可以组合使用或者单板使用。

该实验板包含：

- 用于PIC18微控制器的40针ZIF插座
- 5 V调节器
- 20 MHz振荡器
- 复位键
- 16开关的键盘
- 两个电位计
- 4个LED
- 两行、每行显示20字符的LCD模块

225

- 扬声器
- RC伺服连接器
- RS232接口
- USB连接器
- 数模转换器插座（不带器件）
- I²C串行EEPROM插座（不带器件）



世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源

tyw藏书

- Dallas DS1307实时时钟插座（不带器件）
- Dallas DS18S20温度传感器焊盘（不带器件）
- 内电路编程连接器
- 用于附加电路的原型区域

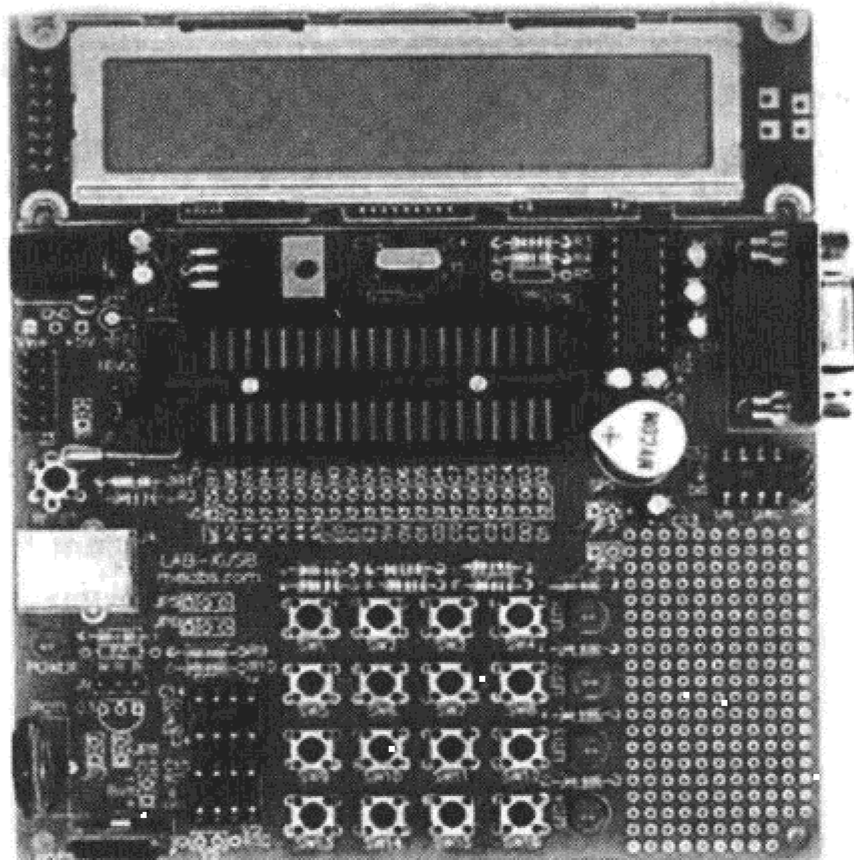


图5-1　LAB-XUSB实验板

2. PICDEM 2 Plus

Microchip公司生产的PICDEM 2 Plus开发板（如图5-2所示），适用于基于PIC18微控制器的项目开发。 226

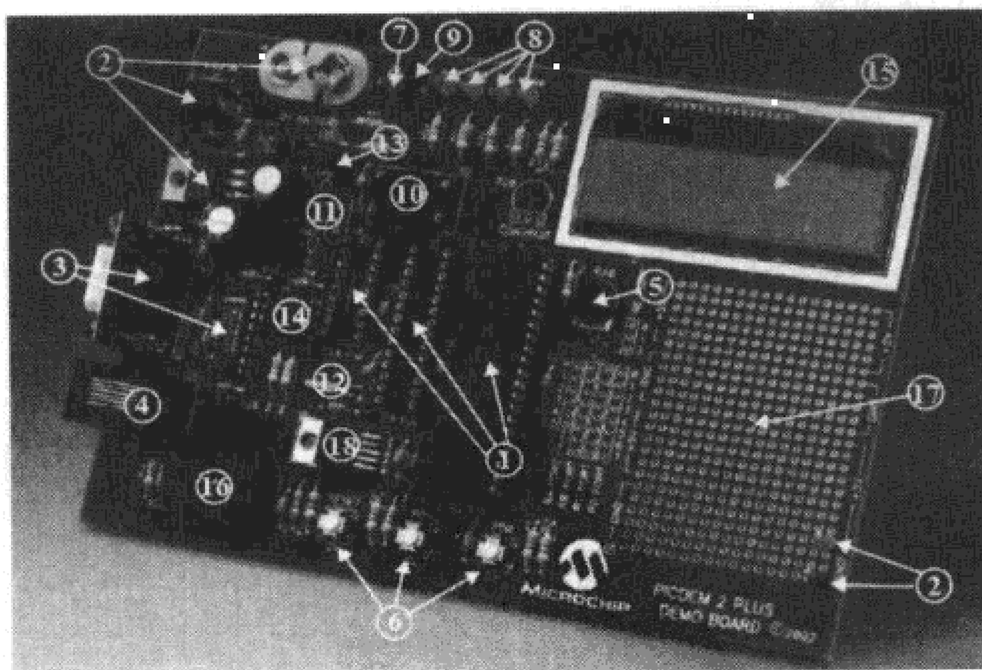


图5-2　PICDEM 2 Plus开发板

该开发板包含：

- 2×16的LCD显示屏

PDG

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

tyw藏书

- PWM驱动的压电式扬声器
- RS232主端口
- 4个LED
- 两个按键开关和主复位键
- 样品PIC18F4520和PIC16F877A闪存微控制器
- MPLAB REAL ICE/MPLAB ICD2连接器
- 所有程序的源代码
- 显示实时时钟和周边环境温度的演示程序
- 充足的原型区域
- 227 9V工作电池或直流电源组

3. PICDEM 4

Microchip公司生产的PICDEM 4开发板（如图5-3所示），适用于基于PIC18微控制器的项目开发。

该开发板包含：

- 3个插座，分别支持8、14和18针的DIP设备
- 板上+5 V电压调节器，输入来自9 V、100 mA的直流/交流适配器
- RS232主端口
- 8个LED
- 2×16的LCD显示屏
- 3个按键开关和主复位键
- 充足的原型区域
- I/O扩展器
- 超级电容器电路
- 228 用于LIN收发器的区域
- 用于电动机驱动器的区域
- MPLAB ICD 2连接器

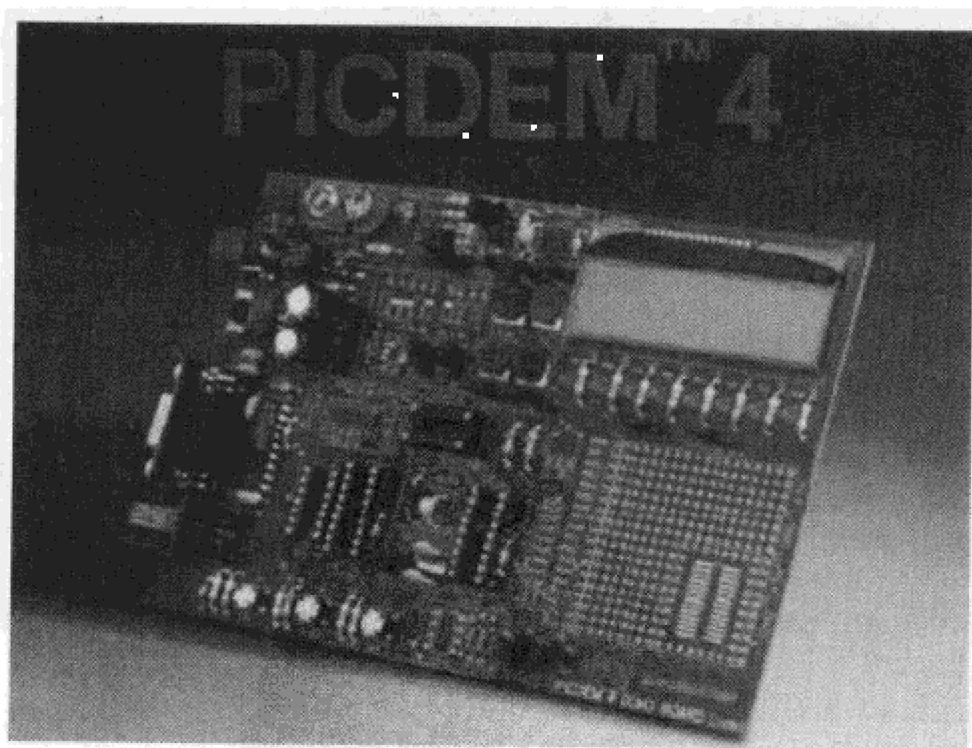


图5-3 PICDEM 4开发板

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

4. PICDEM HPC 开发板

Microchip公司生产的PICDEM HPC开发板（如图5-4所示），适用于引脚数多的PIC18系列微控制器项目开发。

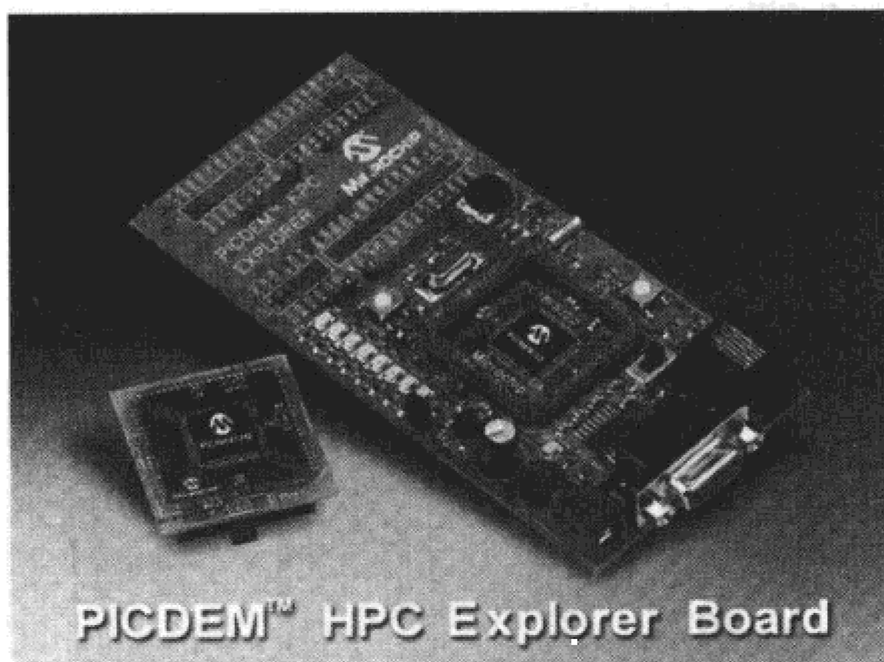


图5-4 PICDEM HPC开发板

该开发板的主要特性有：

- PIC18F8722、128 KB闪存、80针TQFP微控制器
- 支持带插件模块的PIC18J系列设备
- 10 MHz晶振（与内部PLL一起使用，提供40 MHz的操作）
- 电源连接器和可编程的电压调节器，适合2.0 V～5.5 V的工作电压
- 电位计（连接到10位的A/D转换器的模拟输入通道）
- 温度传感器演示单元
- 8个LED（连接到PORTD，禁止跳线）
- RS232端口（9针D型连接器，UART1）
- 复位键
- 用于实时时钟演示的32 KHz晶振

229

5. MK-1 通用PIC开发板

Baji实验室生产的MK-1 通用PIC开发板（如图5-5所示），适用于多达40个引脚的基于PIC18微控制器项目开发。板子有个重要的机制，它允许将任意的外围设备映射到处理器任意引脚上，这使得板子的使用非常灵活。板上带有一小块的面包板，允许用户设计并测试自己的电路。

230

该开发板有以下的特性：

- 板上可选的3.3 V或5 V电源
- 16×2字符的LCD显示（支持8位或4位模式）
- 4位多路复用的7段数码管
- 一排10个的LED（LED可以单独使用）
- 8位的拨码开关
- 带插座的振荡器（易于更换振荡器）
- 集成驱动器的步进电机驱动器
- 使用晶振的I^2C实时时钟，支持电池备份

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

- I^2C温度传感器，精确度为0.5 ℃
- 3个用于直接的模/数转换器开发的分压计
- 4×4矩阵的16按键电话键盘
- 带标准DB9连接器的RS232驱动器
- 插座式的SPI和I^2C EEPROM
- RF发射和接收插座
- IR发射和接收器
- 外部驱动的蜂鸣器
- 上拉电阻
- 交流适配器

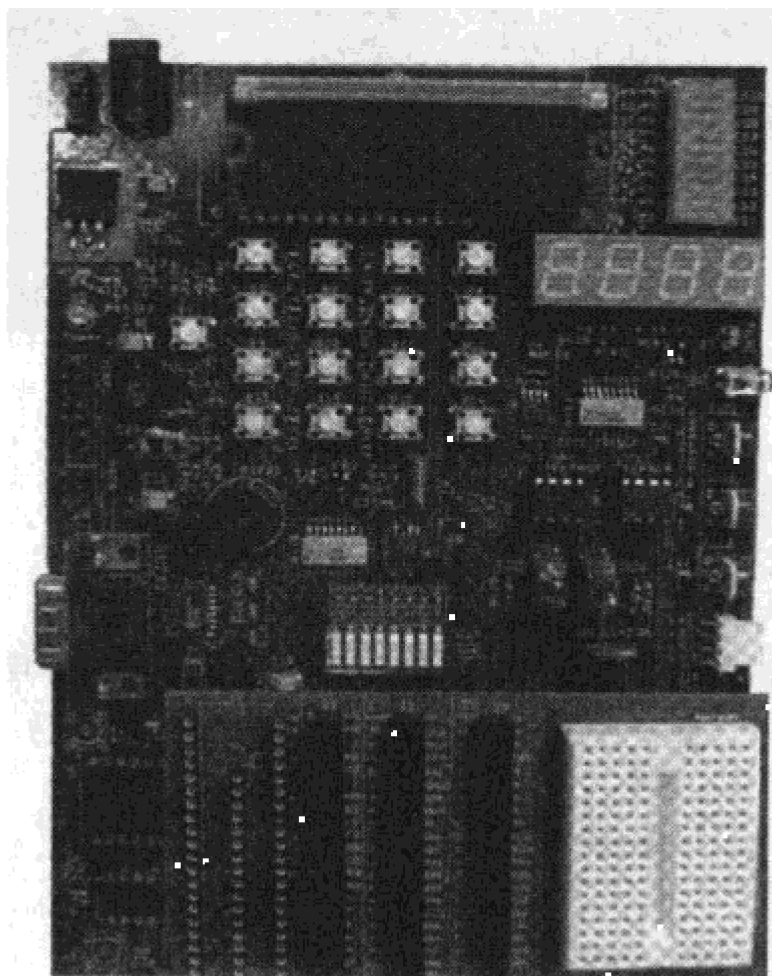


图5-5　MK-1通用PIC开发板

6. SSE452 开发板

由Shuan Shuzi电子实验室生产的SSE452 开发板（如图5-6所示），适用于基于PIC18的微控制器项目开发，尤其是PIC18FXX2系列微控制器的应用开发，也适用于对微控制器进行编程。

该开发板的主要特性有：

231

- 一块适用于28脚或40脚PIC18器件的印刷电路板
- 3个外部中断引脚
- 2个输入捕捉/输出比较/脉冲宽度调制模块（CCP）
- 支持SPI和I^2C功能
- 10位模/数转换器
- RS232连接器
- 两个防抖动的按键开关
- 一个用于数字输入8位DIP开关

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源

tyw藏书

- 4×4键盘连接器
- 带按钮的旋转编码器
- TC77 SPI温度传感器
- EEPROM（24LC04B）
- 2×20总线扩展接口
- ICD2连接器
- 1 Hz～8 MHz的板上多重数字信号
- 可选设备：2×20字符LCD，支持48/28引脚的ZIF插座

232

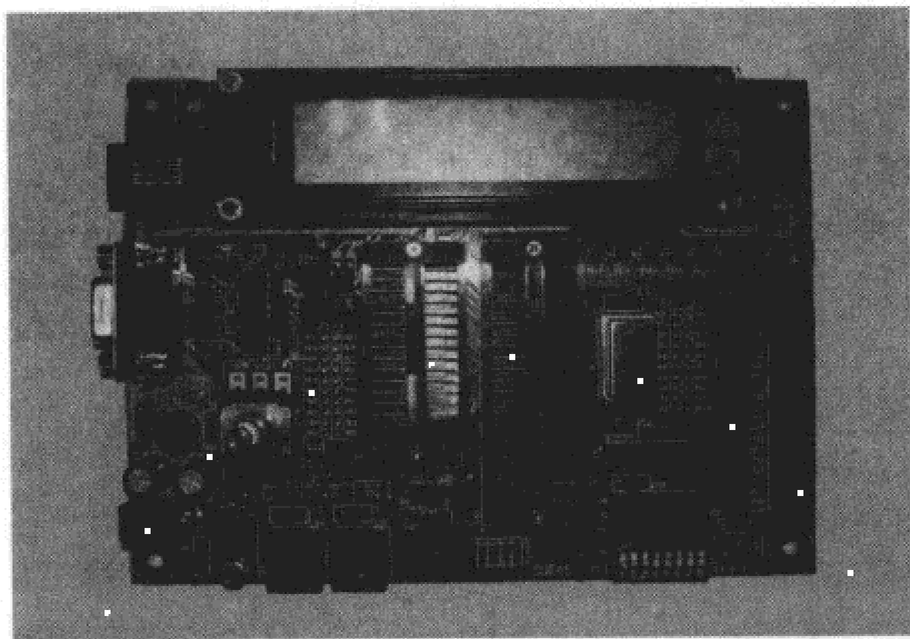


图5-6 SSE452开发板

7. SSE8720开发板

由Shuan Shuzi电子实验室生产的SSE8720开发板（如图5-7所示），适用于基于PIC18微控制器的项目开发。它提供了大量的存储器和I/O接口，板子可用来对微控制器进行编程。

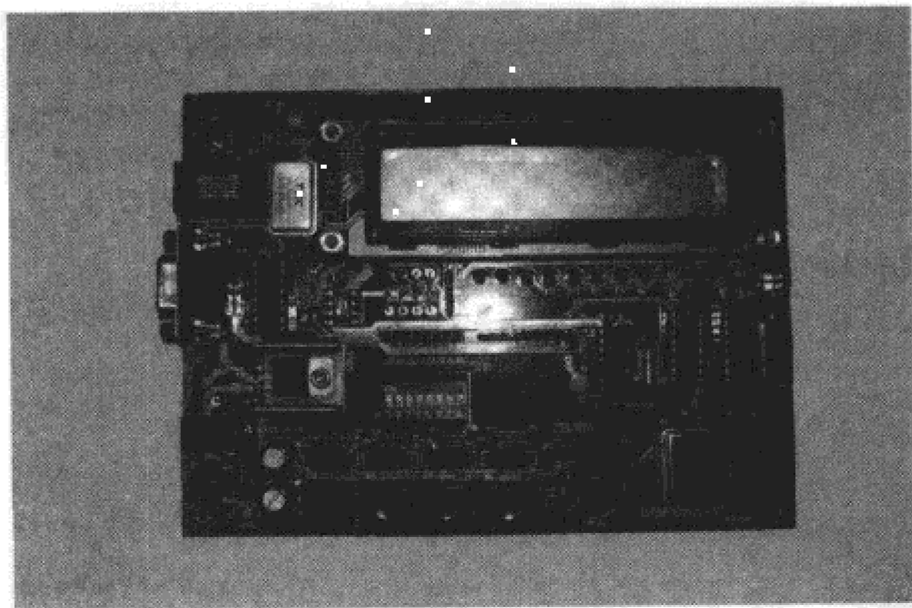


图5-7 SSE8720开发板

该开发板的主要特性有：

- 20 MHz带插座的晶振
- 1个DB9连接器，支持EIA232接口
- 内电路调试器（ICD）连接器

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

- 4个防抖动开关，1个复位按钮
- 4×4键盘连接器
- 1个用于模数转换的电位计
- 8个红色LED

233

- 2×20字符LCD模块
- 24种不同频率的数字信号，从1 Hz到16 MHz
- 板上5 V电压调节器
- 1个带插座的I²C EEPROM
- 兼容SPI的数字温度传感器
- 兼容SPI的实时时钟
- 通过NPN晶体管的CCPI 输出

8. SSE8680开发板

由Shuan Shuzi电子实验室生产的SSE8680开发板（如图5-8所示），适用于基于PIC18微控制器的项目开发。它支持CAN网络，提供大量的存储器和I/O接口。该开发板可用来对微控制器进行编程。

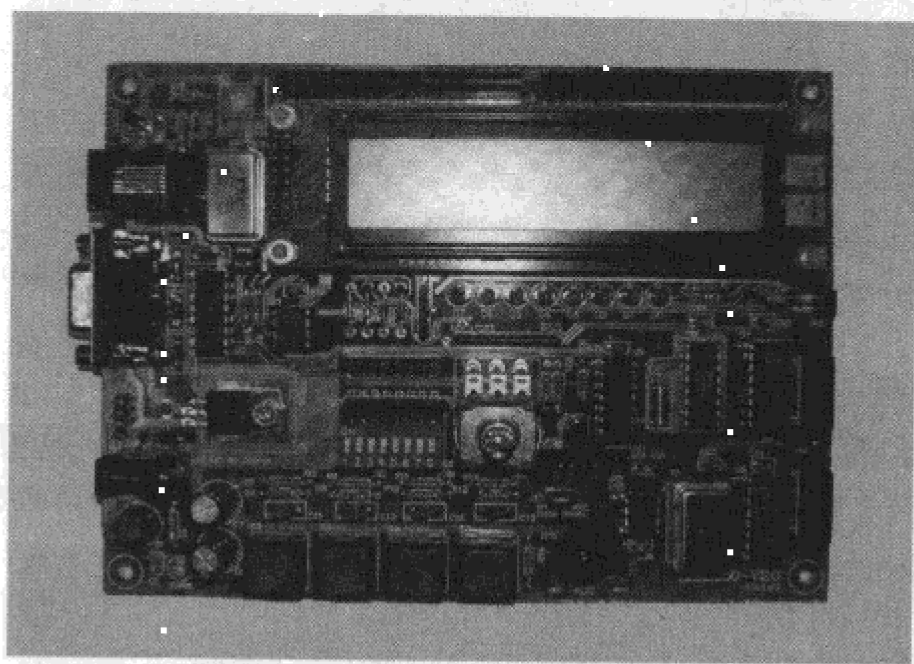


图5-8 SSE8680开发板

该开发板的主要特性有：

- 20 MHz带插座的晶振
- 1个DB9连接器，支持EIA232接口
- 内电路调试器（ICD）连接器
- 4个防抖动开关，1个复位按钮

234

- 4×4键盘连接器
- 1个用于模数转换的电位计
- 8个红色LED
- 2×20字符LCD模块
- 24种不同频率的数字信号，从1 Hz到16 MHz
- 板上5 V电压调节器
- 1个带插座的I²C EEPROM
- 兼容SPI的数字温度传感器

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

- 兼容SPI的实时时钟
- 通过NPN晶体管的CCPI 输出
- 旋转编码器
- CAN收发器

9. PIC18F4520开发工具

由Custom计算机服务公司生产的PIC18F4520开发工具（如图5-9所示），包含了1个C编译器（PCWH）、1块PIC18F4520微控制器原型板、1个内电路调试器和1个编程器。



图5-9　PIC18F4520开发工具

235

该开发工具的主要特性有：

- PCWH编译器
- PIC18F4520原型板
- 模拟板区域
- 93LC56串行EEPROM芯片
- DS1631数字温度计芯片
- NJU6355实时时钟IC，使用32.768 kHz晶振
- 两位7段数码管LED模块
- 内电路调试/编辑器
- DC适配器和线缆

Custom计算机服务公司生产了很多其他基于PIC18微控制器的开发工具和原型板，比如用于CAN、以太网、因特网、USB和串行总线等的开发包。更多的信息，请登录该公司的网页查找。

10. BIGPIC4 开发工具

BIGPIC4是很复杂的开发工具（如图5-10所示），它支持最新的80引脚PIC18微控制器。该开发工具已集成有PIC18F8520微控制器，且工作频率为10 MHz。它包括1个板上USB端口、1个板上编程器和1个内电路调试器。板上的微控制器可非常容易地更换。

236

该开发工具的主要特性有：

- 46个按钮
- 46个LED
- USB接口

PDG

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

- 外部或USB电源供给
- 2个电位计
- 图形LCD
- 2×16文本LCD
- MMC/SD存储卡插槽
- 2个串行RS232端口
- 内电路调试器
- 编程器
- PS2连接器
- 数字温度计芯片（DS1820）
- 模拟输入
- 复位键

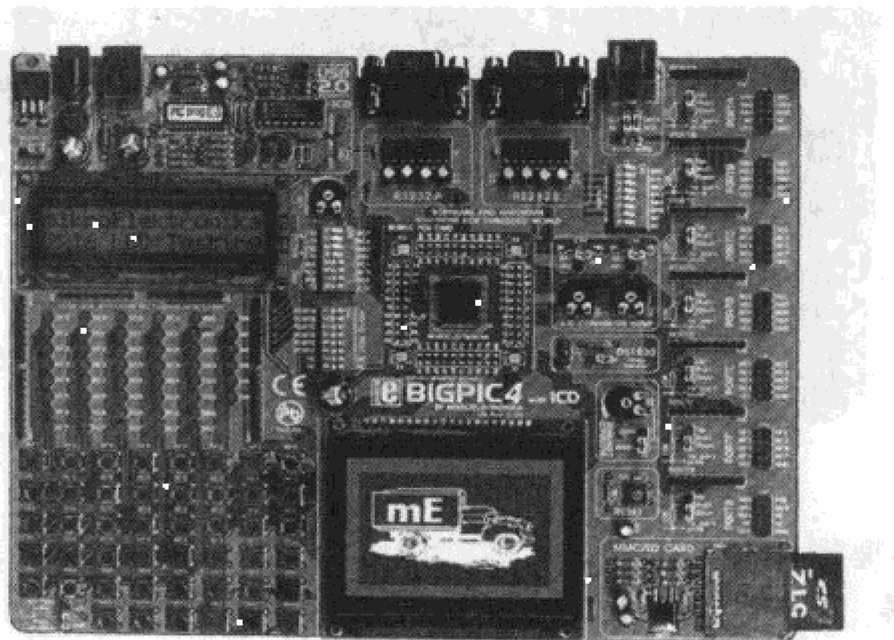


图5-10　BIGPIC4开发工具

在本书的一些项目中，将使用到BIGPIC4。

11. FUTURLEC PIC18F458 训练板

由Futurlec公司（www.futurlec.com）生产的FUTURLEC PIC18F458训练板（如图5-11所示），是功能非常强大的PIC18F458开发工具。该开发工具在出售时已经过组装和测试。它最大的优点就是成本低，还不到45美元。

[237]

该训练板的主要特性有：

- PIC18F458微控制器，使用10 MHz晶振
- RS232通信
- 测试LED
- 可选的实时时钟，带备用电池
- LCD连接
- 可选的RS485/RS422，可选芯片
- CAN和SPI控制器
- I^2C扩展
- 内电路编程器
- 复位键

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

- 扬声器
- 继电器插座
- 连接器的所有端口均有效

238

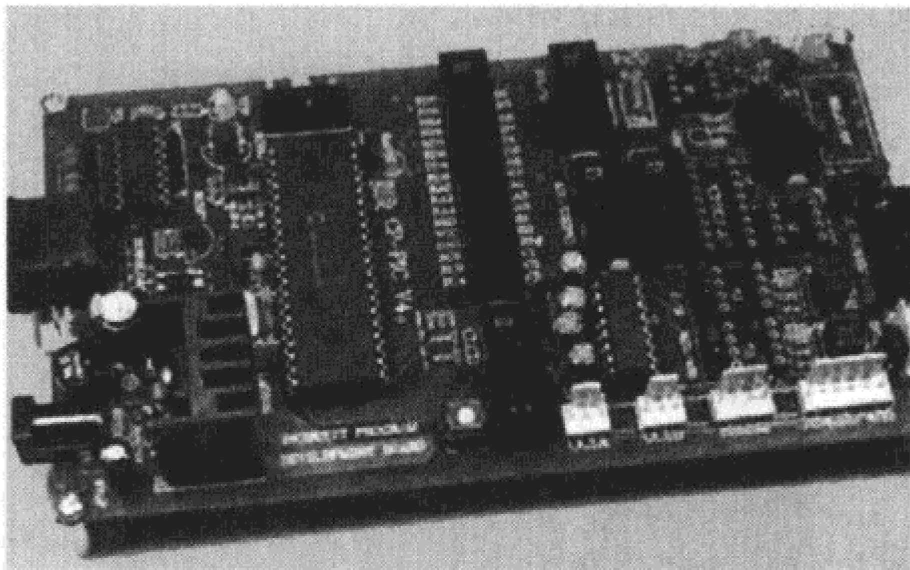


图5-11 FUTURLEC PIC18F458训练板

5.2.2 设备编程器

当写好程序并将其转换成可执行代码后，使用设备编程器可把产生的HEX文件下载到目标微控制器的程序储存区域。设备编程器的类型取决于微控制器的类型。例如，一些设备编程器只支持PIC16系列，一些设备编程器可以同时支持PIC16系列和PIC18系列，而另一些设备编程器则支持其他的微控制器模块（如Intel 8051系列）。

有些微控制器开发工具是提供板上设备编程器的，因此不需要拆除微控制器芯片并插入到专用的编程设备上。本节将描述一些流行的用于PIC18系列微控制器的设备编程器。

1. Forest Electronics USB编程器

由Forest Electronics公司生产的USB编程器（如图5-12所示），可以支持多达40引脚的包括PIC18系列在内大多数PIC微控制器。将USB编程器连接到计算机的USB端口，可从中获取电源供给。

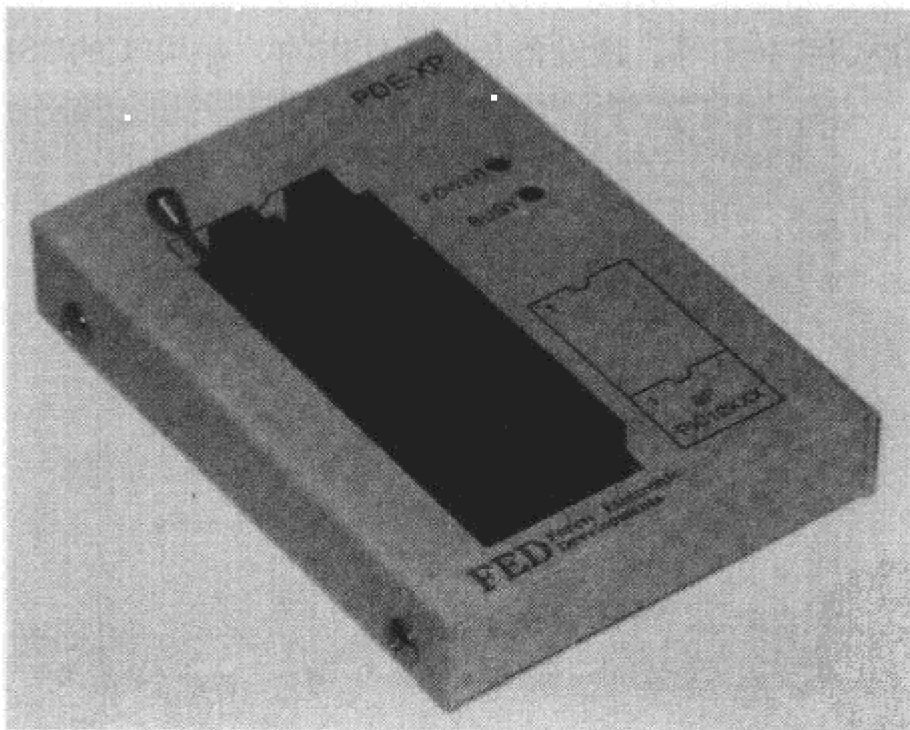


图5-12 Forest Electronics USB编程器

239

PDG

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

2. Mach X编程器

由Custom计算机服务公司生产的Mach X编程器（如图5-13所示），支持PIC12、PIC14、PIC16和PIC18系列的微控制器，引脚数从8到40不等。该编程器也可以从微控制器中读出程序，然后转换成HEX文件。同时，它还支持内电路调试。



图5-13　Mach X编程器

3. Melabs U2编程器

由mikroEngineering实验室公司生产的Melabs U2编程器（如图5-14所示），支持大多数的PIC微控制器芯片，引脚数从8到40不等。该设备是基于USB的，并从PC的USB端口获取电源供给。

240

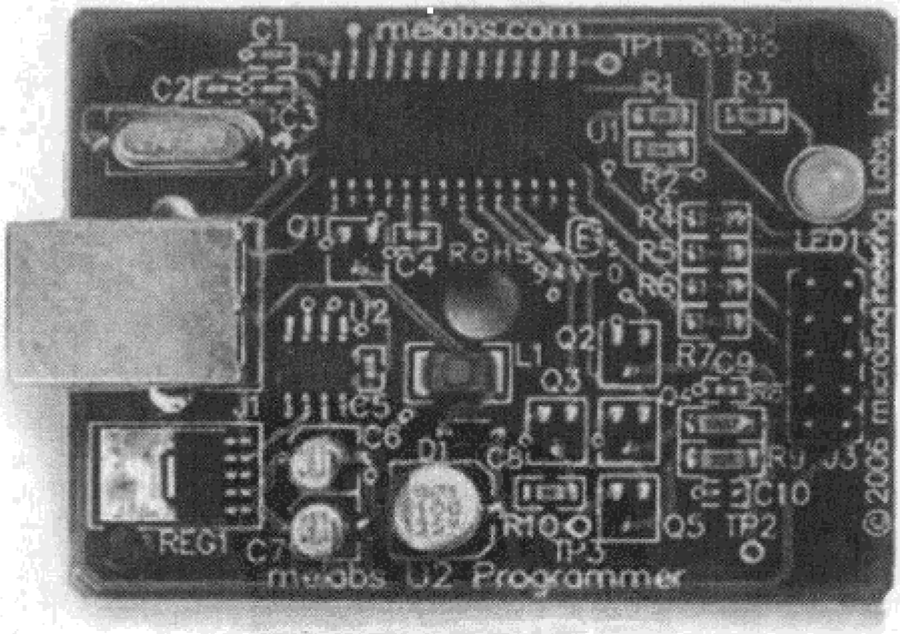


图5-14　Melabs U2编程器

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源

tyw藏书

4. EasyProg PIC编程器

EasyProg PIC编程器是一款低成本的编程器（如图5-15所示），支持多达40引脚的PIC16和PIC18系列微控制器。它通过9针的串行电缆连接到PC。 241

5. PIC Prog Plus编程器

PIC Prog Plus是又一款低成本的编程器（如图5-16所示），支持大部分的PIC微控制器。该编程器从外部12V直流电源获取工作电压。

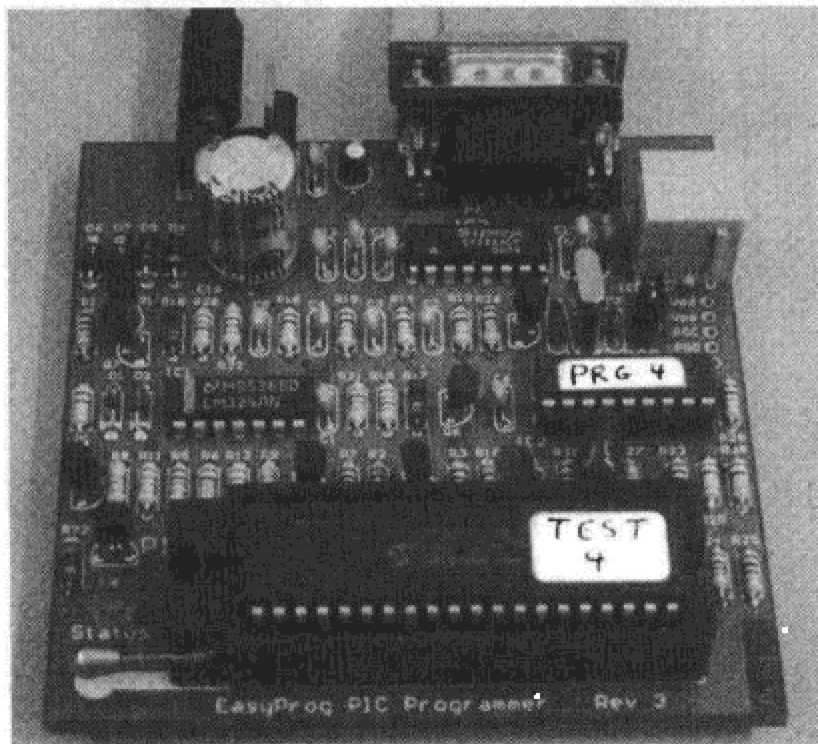


图5-15 EasyProg PIC编程器

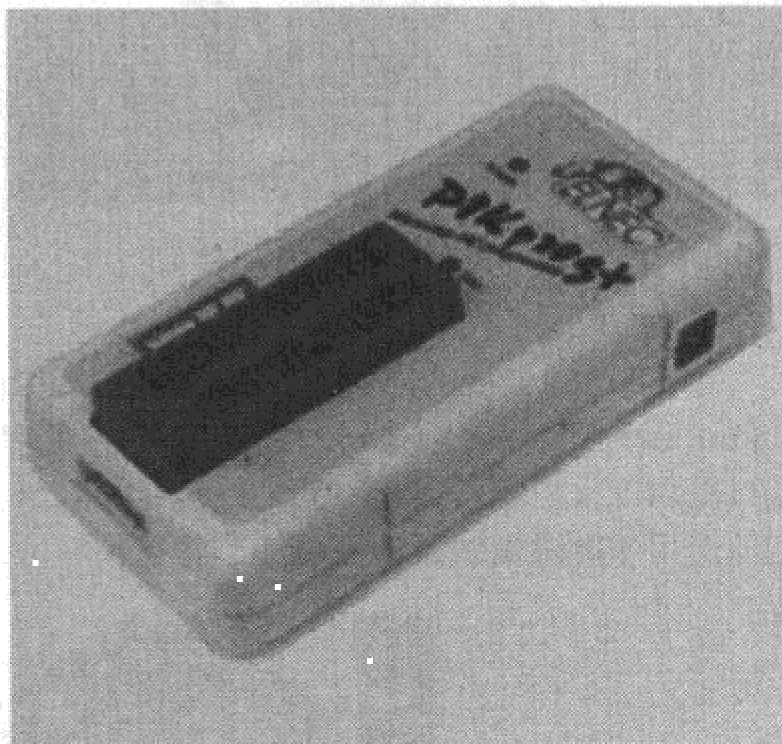


图5-16 PIC Prog Plus编程器

5.2.3 内电路调试器

内电路调试器是连接在PC和目标微控制器测试系统之间的硬件。它使得实时应用的调试变得简单而快捷。使用内电路调试，就是让监视程序运行在测试电路的PIC微控制器中。程序员可以在PIC上设置断点，在实际设备中运行代码、单步运行程序、检查变量和储存器等。如有必要，还可以改变它们的值。内电路调试器在调试过程中会用到目标微控制器的一些存储空间和I/O端口。有些内电路调试器只能调试汇编程序。而另一些功能更强大的调试器则可以调试高级语言程序。 242

本节将讨论一些在PIC微控制器系统应用中比较流行的内电路调试器。

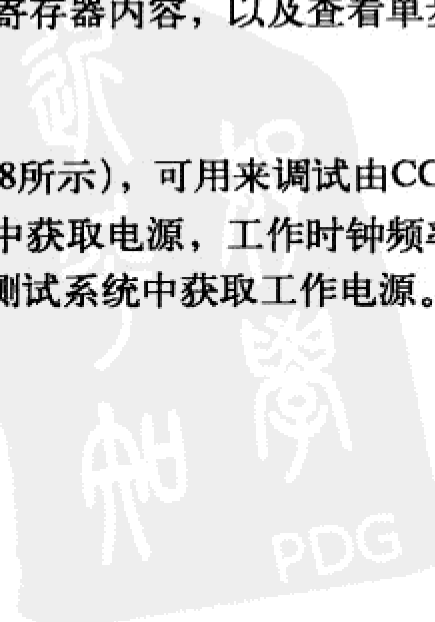
1. ICD2

ICD2是Microchip公司生产的一款低成本的内电路调试器（如图5-17所示），可以支持大部分的PIC微控制器系统。使用ICD2，可以将程序下载到目标微控制器芯片，并进行实时运行。该调试器既支持汇编语言，也支持C语言程序。

ICD2可以通过串行RS232或USB接口连接到PC。该设备就像一个PC与测试系统之间的智能接口，允许程序员设置断点、检查测试系统、查看断点处的变量和寄存器内容，以及查看单步运行程序等。它也可以用来对目标PIC微控制器进行编程。

2. ICD-U40

Custom计算机服务公司生产的ICD-U40内电路调试器（如图5-18所示），可用来调试由CCS C编译器开发的程序。该设备通过USB接口接到PC上，并从USB端中获取电源，工作时钟频率为40 MHz。该公司还生产有串口版本的调试器ICD-S40，它从目标测试系统中获取工作电源。 243



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

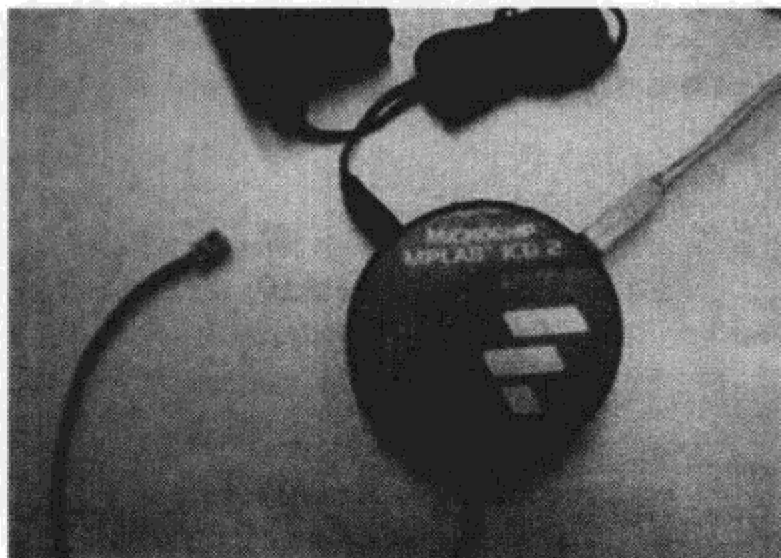


图5-17 ICD2内电路调试器

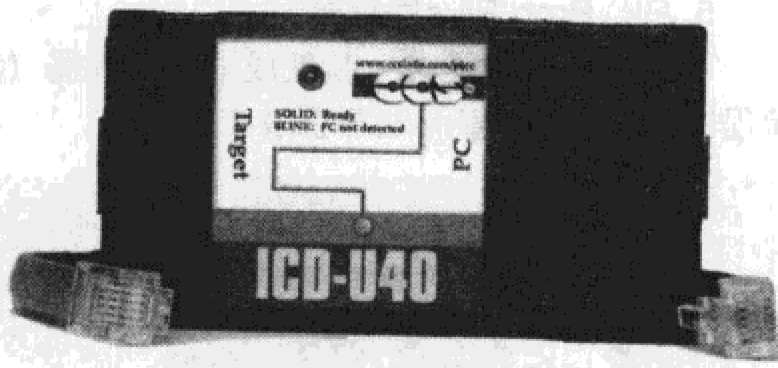


图5-18 ICD-U40内电路调试器

3. PICFlash 2

由mikroElektronika公司生产的PICFlash 2内电路调试器（如图5-19所示），可用来调试由mikroBasic、mikroC或mikroPascal语言开发的程序。该设备通过USB接口连接到PC上，并从USB端口获取工作电源，因此不需要连接外部电源。PICFlash 2包含在BIGPIC4开发工具包中。关于如何使用内电路调试器的更多细节将在稍后讨论。

244



图5-19 PICFlash 2内电路调试器

5.2.4 内电路模拟器

内电路模拟器（ICE）是最早且功能最强大的微控制器系统调试设备之一。它也是唯一使用内部处理器代替目标系统微控制器的工具。就像所有的内电路调试器一样，模拟器的主要功能是目标访问能力，即检查和修改寄存器、内存和I/O端口内容的能力。因为使用了模拟器代替系统的CPU，所以它并不需要目标系统中的工作CPU。这就使得内电路模拟器成为目前用于修正新系统或完善旧系统的最有效工具。

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com
tyw藏书

一般而言，每个微控制器系列都有它们自己的内电路模拟器。例如，PIC16专用的内电路模拟器就不可以在PIC18上使用。另外，内电路模拟器的成本通常很高。为了降低成本，模拟器制造商都提供一个基板，可用于指定系列的大多数微控制器（如PIC18微控制器系列），并为特定的微控制器提供可用的插卡。例如，要模拟同一个系列的新型微控制器，只需要购买特定的插卡就可以。

现在市场上有几种内电路模拟器。下面介绍4款比较流行的模拟器。

1. MPLAB ICE 4000

由Microchip公司生产的MPLAB ICE 4000内电路模拟器（见图5-20），可用来仿真PIC18系列的微控制器。它包含有一个模拟头，通过柔性电缆与特定微控制器的设备适配器相连。模拟头的另一端通过并行端口或者USB端口与PC相连。用户可以设置无限个断点来观察寄存器的值。

2. RICE3000

245

由Smart通信有限公司生产的RICE3000（见图5-21），是一款功能强大的内电路模拟器，用于PIC16系列和PIC18系列微控制器。它提供一个基础单元，并配有适于PIC微控制器系列不同型号的插卡。该模拟器提供了全速的实时模拟，运行速度达40 MHz，可以观察浮点数变量和复数变量（如数组和数据结构），还在汇编和高级语言中提供源代码级调试和符号调试。



图5-20 MPLAB ICE 4000

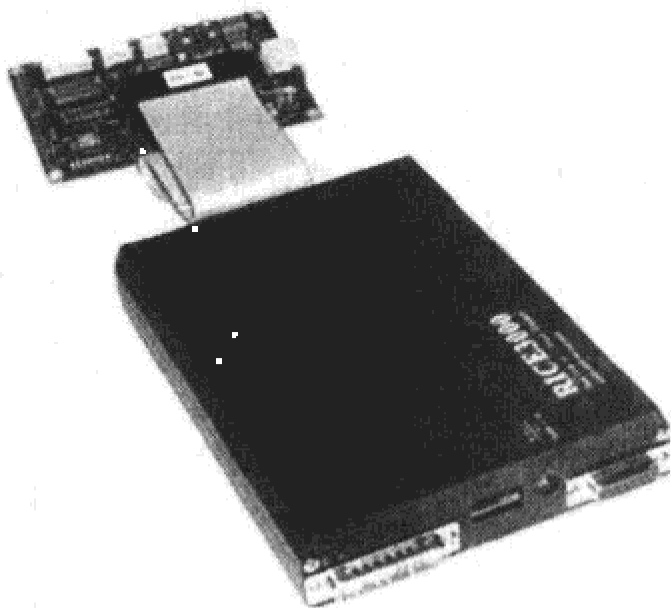


图5-21 RICE3000内电路模拟器

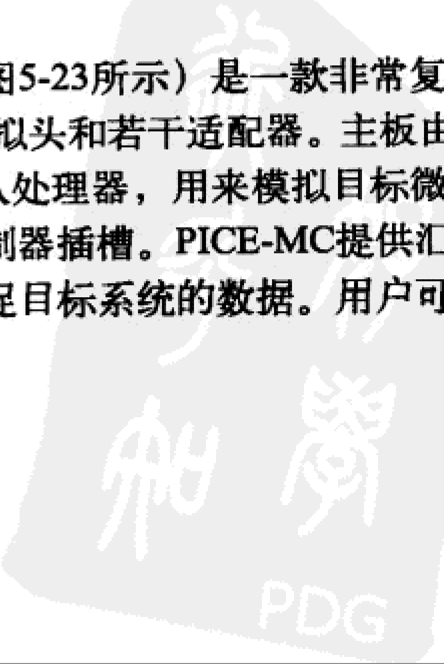
3. ICEPIC 3

由RF资讯公司生产的ICEPIC 3（如图5-22所示）是一款模块化的内电路模拟器，支持PIC12/16和PIC18系列的微控制器。它通过USB端口连接到PC。该模拟器包括一块母板，并根据微控制器的类型配有额外的子板。子板通过设备适配器与目标系统相连。另外，跟踪调试板也可以用来捕捉和分析执行地址、操作码和外部存储器的读写操作。

4. PICE-MC

246
~
247

由Phyton公司生产的PICE-MC（如图5-23所示）是一款非常复杂的模拟器，它支持绝大部分的PIC微控制器，包含1块主板、1个模拟头和若干适配器。主板由模拟器逻辑、存储器和连到PC的接口部分组成。模拟头包含有1个从处理器，用来模拟目标微控制器。适配器是一些机械部件，用在物理连接到目标系统的微控制器插槽。PICE-MC提供汇编语言和高级语言的源代码级调试，并提供了很大的存储空间来捕捉目标系统的数据。用户可以设置很多断点，显示和修改程序和数据存储器内容。


PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书



图5-22 ICEPIC 3内电路模拟器

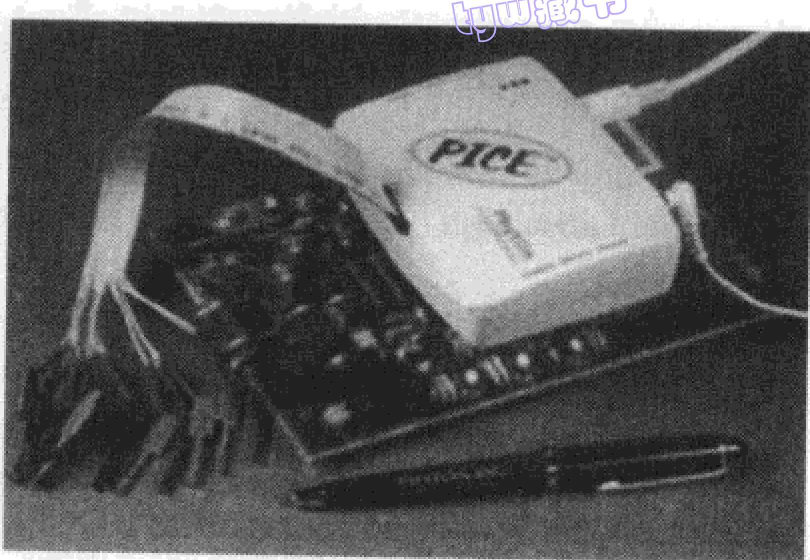


图5-23 PICE-MC内电路模拟器

5.2.5 面包板

在制作电子电路的时候，需要按照相关的电路图连接各个部件，通常将各组成部件焊接在印刷电路板（PCB）上。当电路已通过测试并工作正常，或者要求电路是永久性制作时，这种方法是合适的。但是，对于某些应用的PCB板设计（如仍然需要开发的电路），这个方法就不够经济了。

取而代之地，当一个电路还处于开发阶段时，各元件通常安装在非焊接的面包板上。常见的面包板（如图5-24所示），由很多行列分布的插孔组成，这样集成电路和其他元件可以容易地安装上去。这些孔都是有弹性的，元件插脚可以在板上得到很好的固定。市面上有很多不同规格尺寸的面包板，适合各种复杂的电路。也可以把几个面包板组合在一起，形成更大的面包板，以适用于更复杂的电路。图5-25给出了图5-24中面包板的内部走线。

248

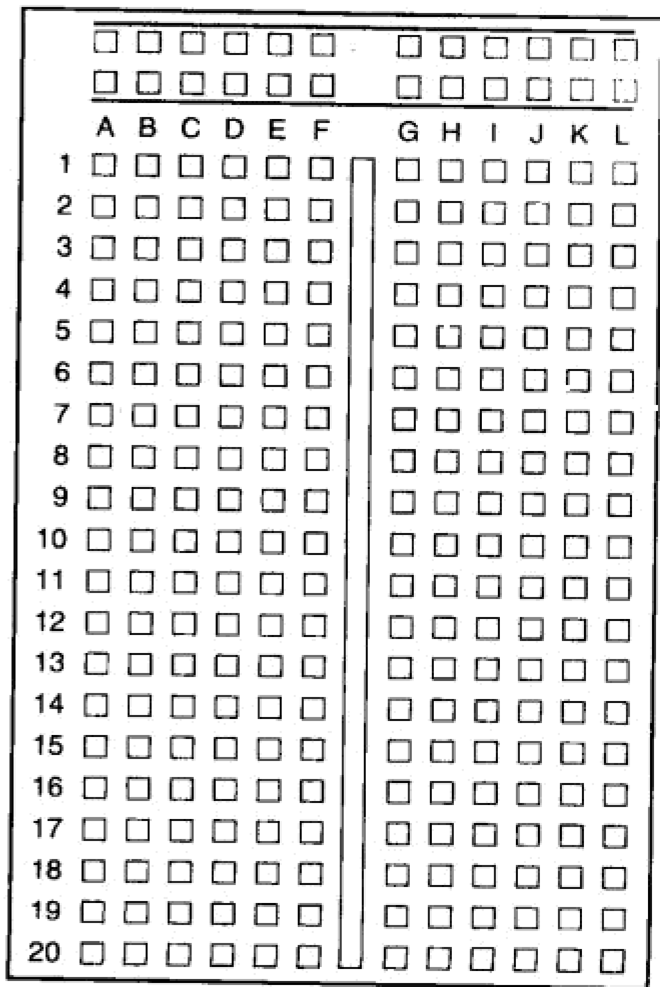


图5-24 常见面包板的布局图

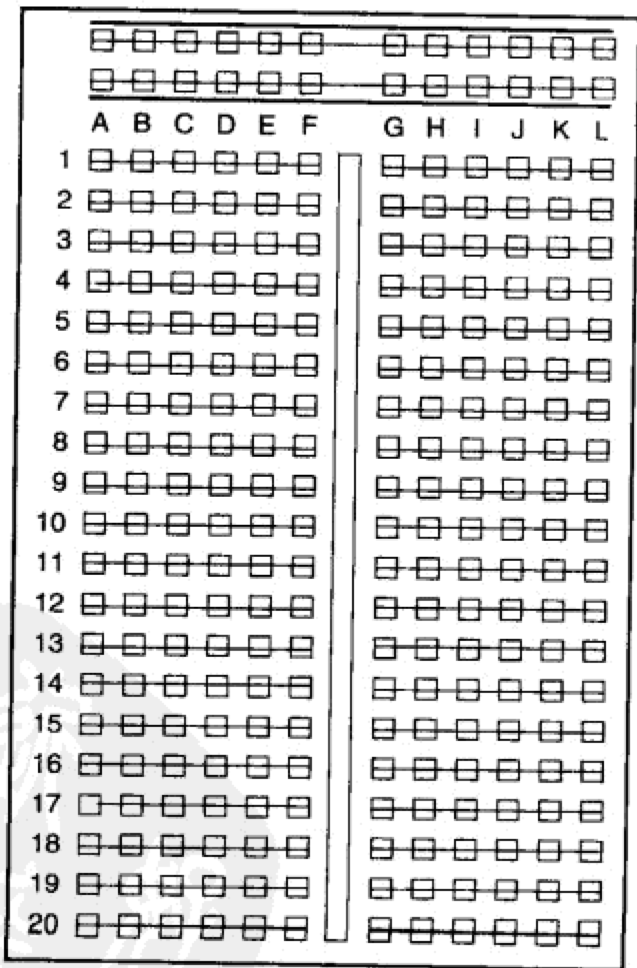


图5-25 面包板的内部布线

PDG

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

面包板的左右两部分是完全分开的。从1到20行，每行从A到F都是连接到一起的，同样地，从G到L也是连接在一起的。集成电路块安放在面包板上，一组引脚插在面包板左半部分，另一组引脚则插在板子的右半部分。板子最顶部的那两行插孔，通常用于连接电源和接地。通常，使用标准线缆或者硬线插入小孔来连接各个元件。

249

图5-26给出了将两个集成电路、一些电阻和电容安装在面包板上的示意图。

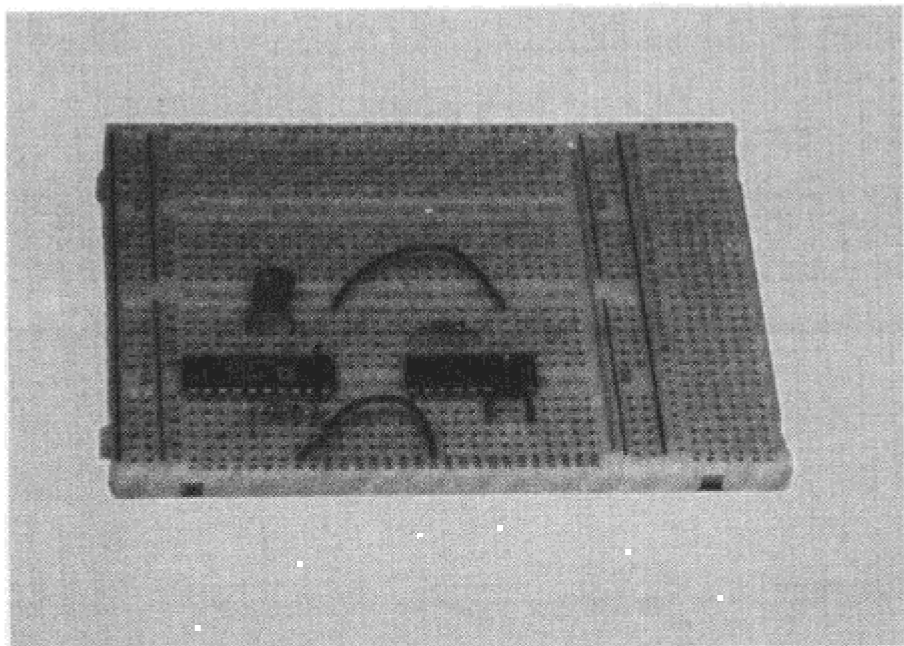


图5-26 安装有元器件的面包板照片

使用面包板的好处就是，容易快速地修改电路，并且不需要把元件焊接起来再进行测试。当电路测试通过且工作正常后，还可以移除元件，将面包板再次用于其他的项目。

5.3 mikroC 集成开发环境（IDE）

在本书中，讨论的是mikroElektronika公司开发的mikroC编译器。在使用之前，需要了解mikroC集成开发环境（IDE）是怎样建立、编写、编译和仿真一个mikroC程序的。本章将详细介绍mikroC IDE的操作。

在mikroElektronika公司的网站（www.mikroe.com）上可以找到免费的限制程序大小为2 KB的mikroC IDE，这对于大部分小型和中型的应用已经足够了。当然，也可以购买使用许可证，把受限的版本升级成完整版。完全版本适用于任何大小和复杂度的项目。

在安装了mikroC IDE之后，桌面上会默认出现一个新图标。双击该图标启动IDE。

5.3.1 mikroC IDE 界面

在双击打开了IDE之后，弹出默认的界面，如图5-27所示。屏幕界面分为4个区域：左上部分、左下部分、中间部分和底部区域。

1. 左上部分

左上部分是代码浏览（Code Explorer）区域，用来显示所有源代码中声明过的项。在图5-28的例子中，在Functions项下列出了`main`函数，在main项下又列出了变量`Sum`和`i`。

250~251

Code Explorer还有两个附加的标签。如图5-29所示，QHelp标签列出了全部可用的内置函数和库函数，以供快速引用。

Keyboard标签列出了所有可用的mikroC IDE快捷键（如图5-30所示）。

PDG

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

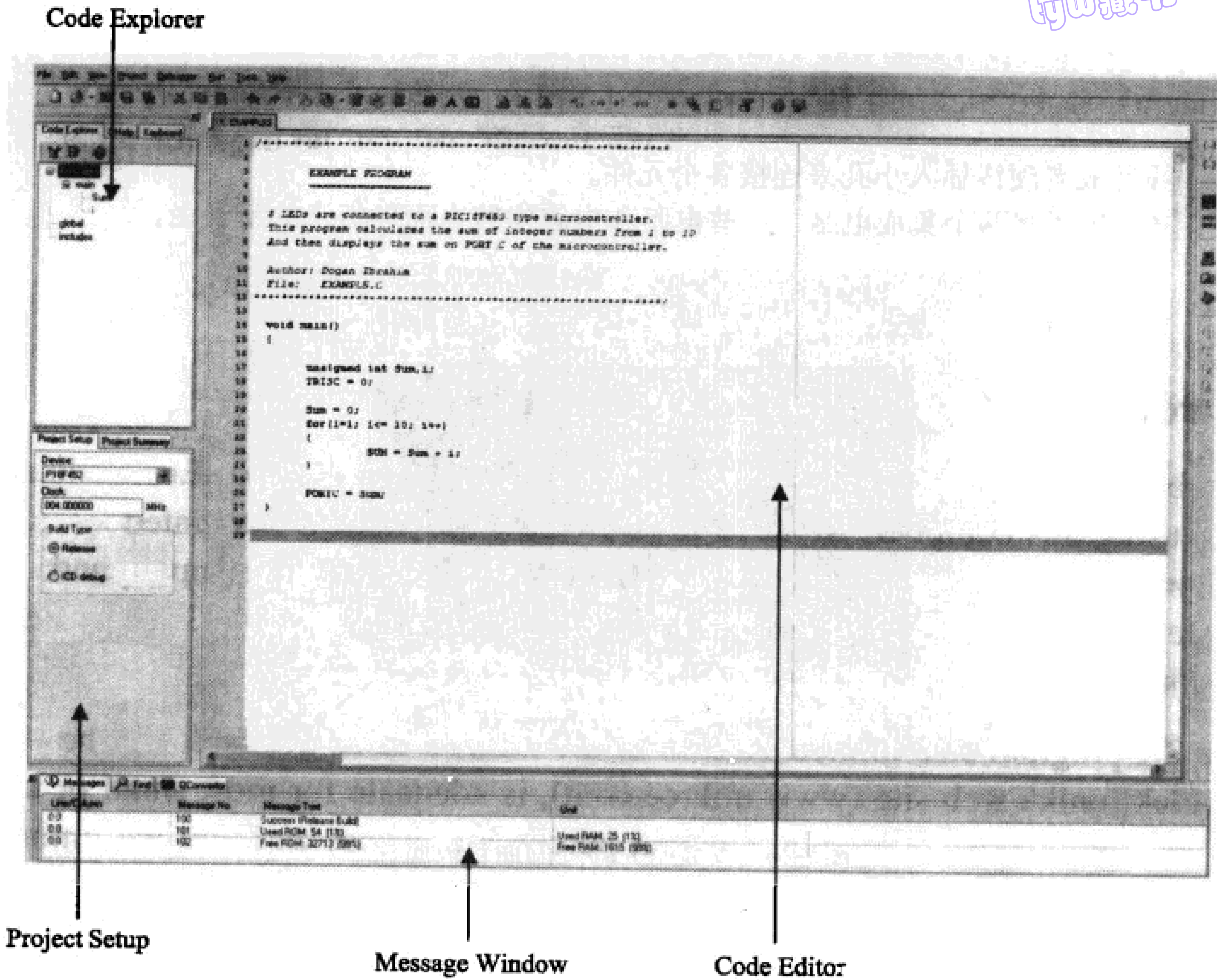


图5-27　mikroC集成开发环境界面

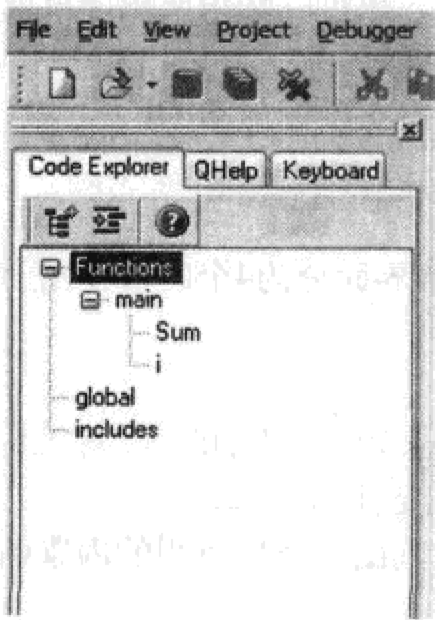


图5-28　代码浏览器布局

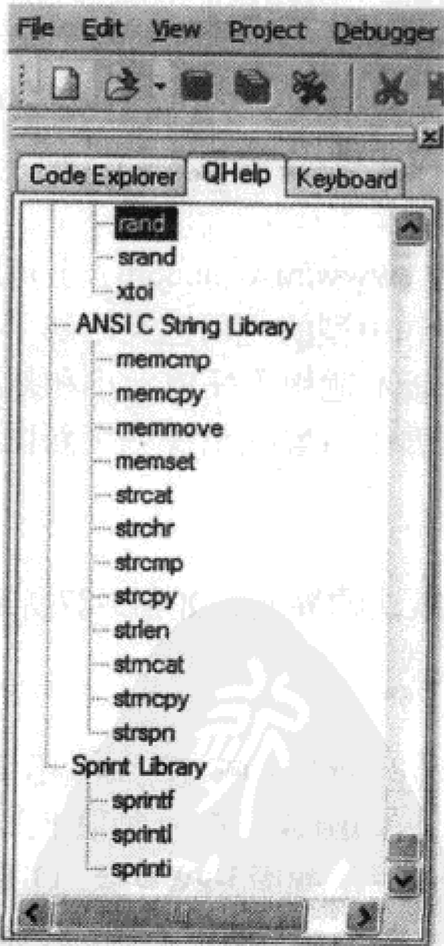


图5-29　QHelp布局

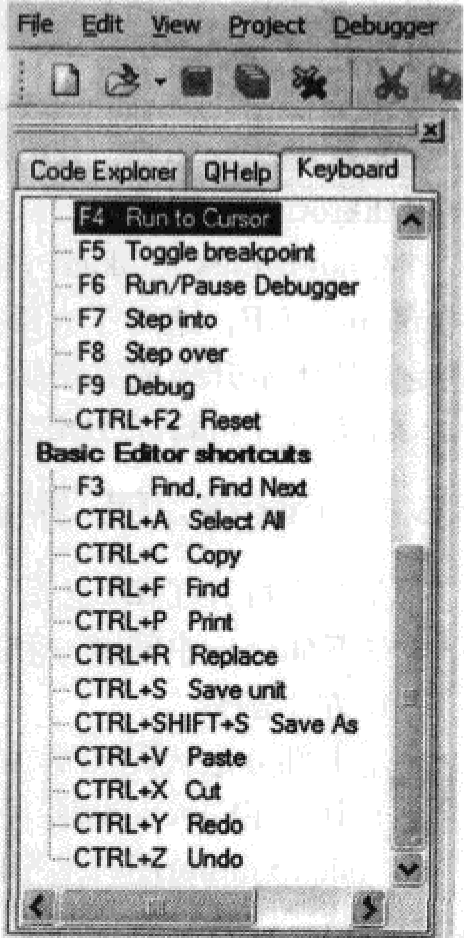


图5-30　Keyboard布局

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

2. 左下部分

左下部分是项目设置（Project Setup）区域（如图5-31所示），用来指定微控制器的类型、时钟速率和构建类型。构建类型可以是Release模式，即正常的编译器模式；也可以是ICD调试模式，当使用内电路调试器对程序进行调试时选用该构建模式。

在项目建立区域有一个项目总汇（Project Summary）标签，列出了项目中所有用到的文件类型，如图5-32所示。

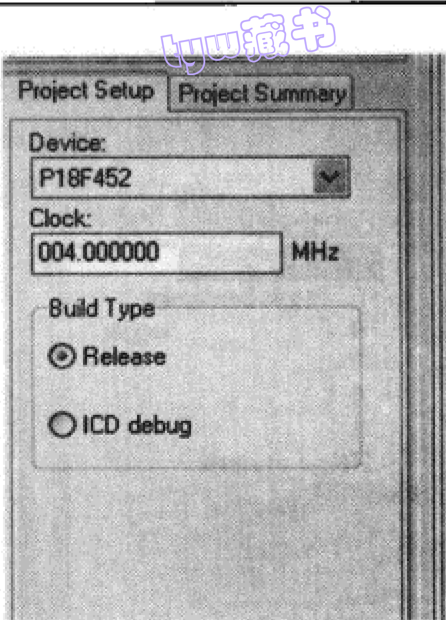


图5-31 Project Setup布局

252

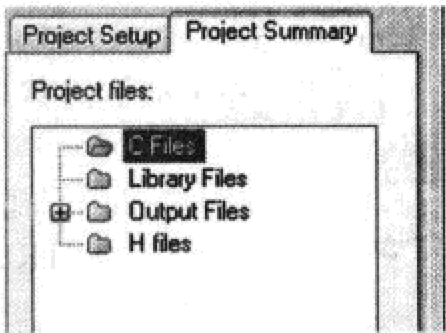


图5-32 Project Summary布局

253

254

3. 中间部分

中间部分是代码编辑（Code Editor）区域，这是一个高级文本编辑器。程序在这个区域进行编写。Code Editor支持：

- 代码辅助（Code Assistant）
- 参数辅助（Parameter Assistant）
- 代码模板（Code Template）
- 自动更正（Auto Correct）
- 书签（Bookmarks）

Code Assistant在编写程序的时候非常有用。键入标识符前几个字母，再按下CTRL+SPACE组合键，就会列出所有可用的带有这几个字母的标识符。例如，在图5-33中，要找标识符strlen，则首先键入字符str，然后按下CTRL+SPACE组合键。就可以从列出的匹配字中找到strlen，然后用键盘方向键选择，并按ENTER键。

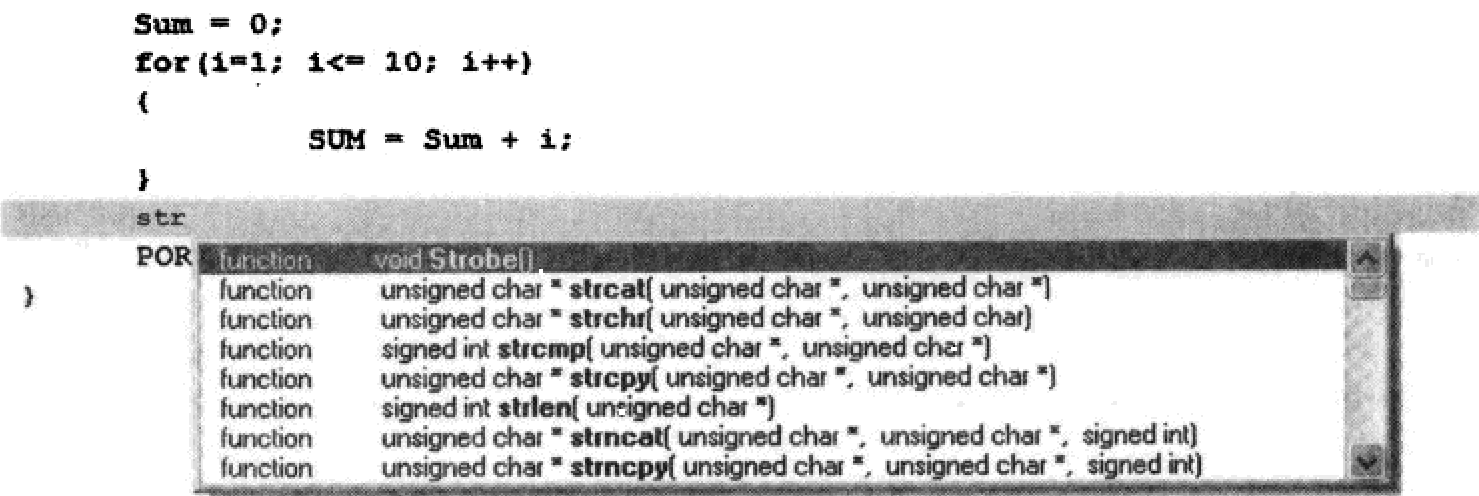


图5-33 使用Code Assistant

当在函数或者过程名后面打开了一个括号时，Parameter Assistant就会被激活。期望的参数将会在括号上面的小窗口中列出。在图5-34中，键入了函数strlen，在添加括号的时候，unsigned char *s就出现在小窗口内。

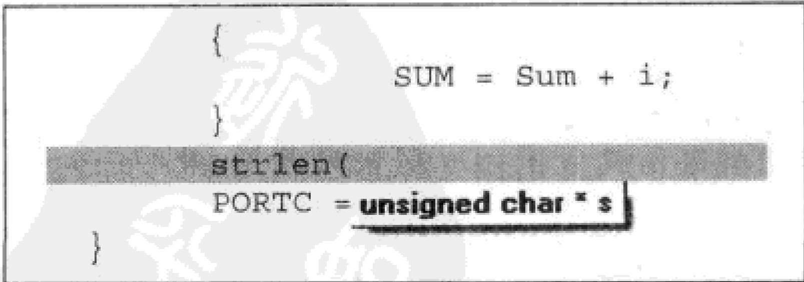


图5-34 使用Parameter Assistant

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

255 Code Template用于在程序中生成代码。比如，在图5-35中，只要键入switch，并按下CTRL+L组合键，就可自动地为switch语句生成一段代码。也可以依次选择Tools→Options→Auto→Complete来添加自己的模板。可用的模板有array、switch、for和if等。

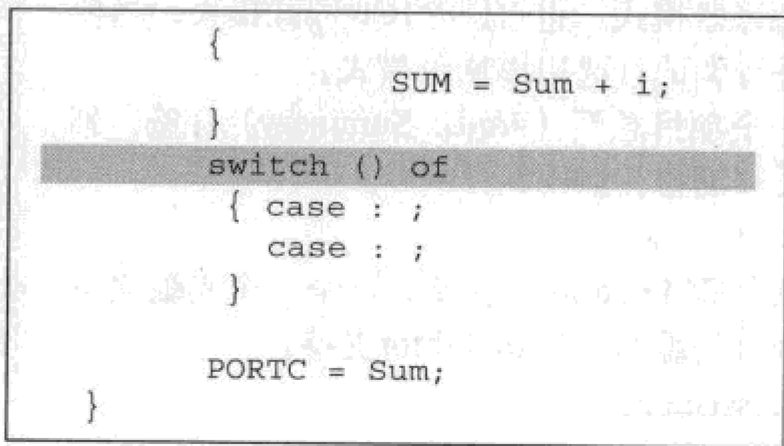


图5-35 使用Code Template

256~257 Auto Correct功能可以自动地修正输入错误。也可以依次选择Tools→Options→Auto Correct Tab，在列表中添加一些新的词语。

Bookmarks使得在大型代码中的导航更加简单。可以按组合键CTRL+SHIFT+数字，来设置一个书签。按组合键CTRL+数字，就可以跳到数字所指定的书签位置。

4. 底部区域

底部区域又被称作消息窗口（Message Window）。它由3个标签组成：Message、Find和QConverter。Message标签用来显示编辑错误和警告信息。双击该信息，可以将出错的代码行高亮显示。HEX文件只有在源文件没有任何错误的情况下才能生成。在图5-36中，Message窗口给出了编译成功的信息。QConverter标签用来将十进制数字转换成二进制或者十六进制数，也可以进行相反方向的转换。

Messages | Find | QConvertor

Line/Column	Message No.	Message Text	Unit
0:0	100	Success (Release Build)	
0:0	101	Used ROM: 54 (1%)	Used RAM: 25 (1%)
0:0	102	Free ROM: 32713 (99%)	Free RAM: 1615 (99%)

图5-36 编辑成功的信息显示

5.3.2 创建和编译新文件

mikroC文件是以项目形式组织的，单个项目的所有文件都存储在同一个文件夹里。项目文件的扩展名默认为.ppc。项目文件包括项目名称、目标微控制器设备、设备配置标志、设备时钟和源文件的列表及其路径。C语言的源文件扩展名为.c。

下面的例子将说明如何一步一步地创建和编译程序源文件。

例5.1 编写C程序，计算整型数值1～10的总和，并将结果发送到PIC18F452微控制器的PORTC。假设8个LED通过限流电阻连接到微控制器的PORTC。绘制电路图，并列出创建和编译程序所需的步骤。

258

解 图5-37给出了该项目的电路图。LED连接到PORTC上，使用390 Ω的限流电阻。微控制器工作频率为4 MHz。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

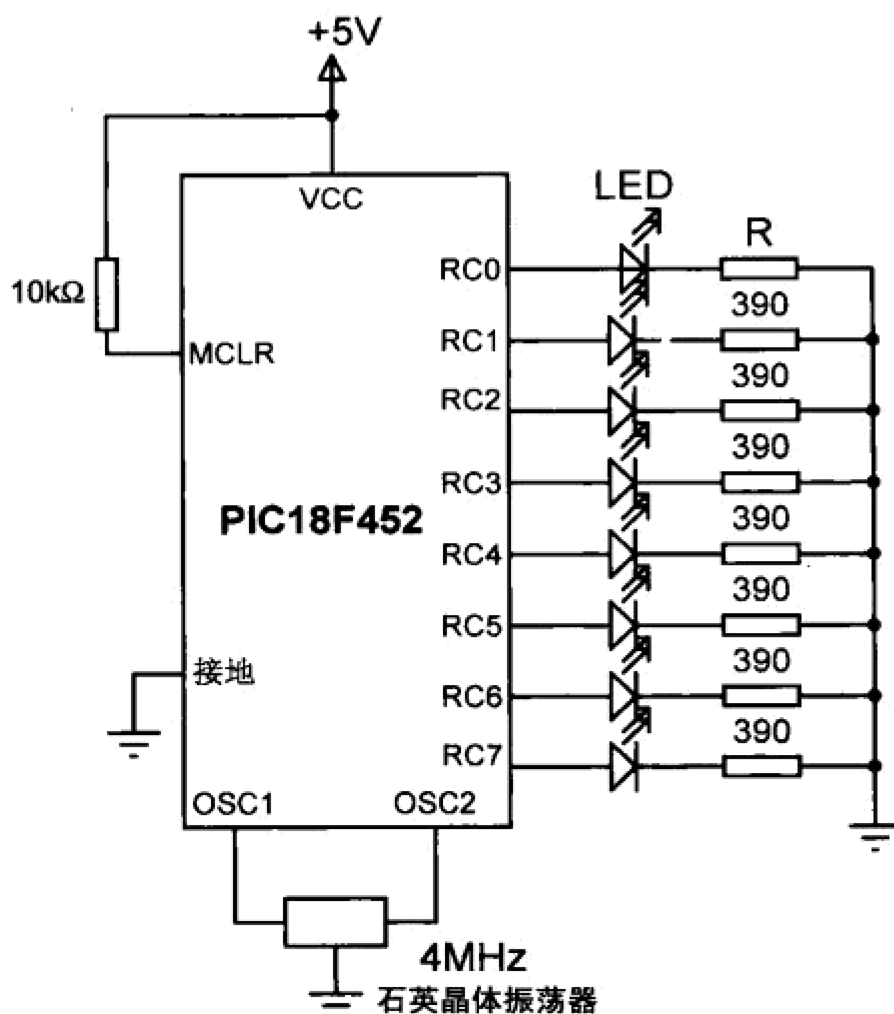


图5-37 项目的电路图

项目的创建和编译步骤如下。

步骤1 双击mikroC图标，启动IDE。

步骤2 创建一个名字为EXAMPLE的新项目。点击Project→New Project，并填写表格，选择设备类型、时钟和配置标志，如图5-38所示。

259
~
260

步骤3 在IDE的Code Editor区域录入下面的程序代码：

```
/**********************************************************************
                         EXAMPLE PROGRAM

     8 LEDs are connected to a PIC18F452 type microcontroller.
     This program calculates the sum of integer numbers from 1 to 10
     And then displays the sum on PORTC of the microcontroller.

     Author: Dogan Ibrahim
     File:   EXAMPLE.C

**********************************************************************/
void main()
{

     unsigned int Sum,i;
     TRISC = 0;

     Sum = 0;
     for(i = 1; i<=I0; i++)
     {
               Sum = Sum + i;
     }

     PORTC = Sum;
}
```

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

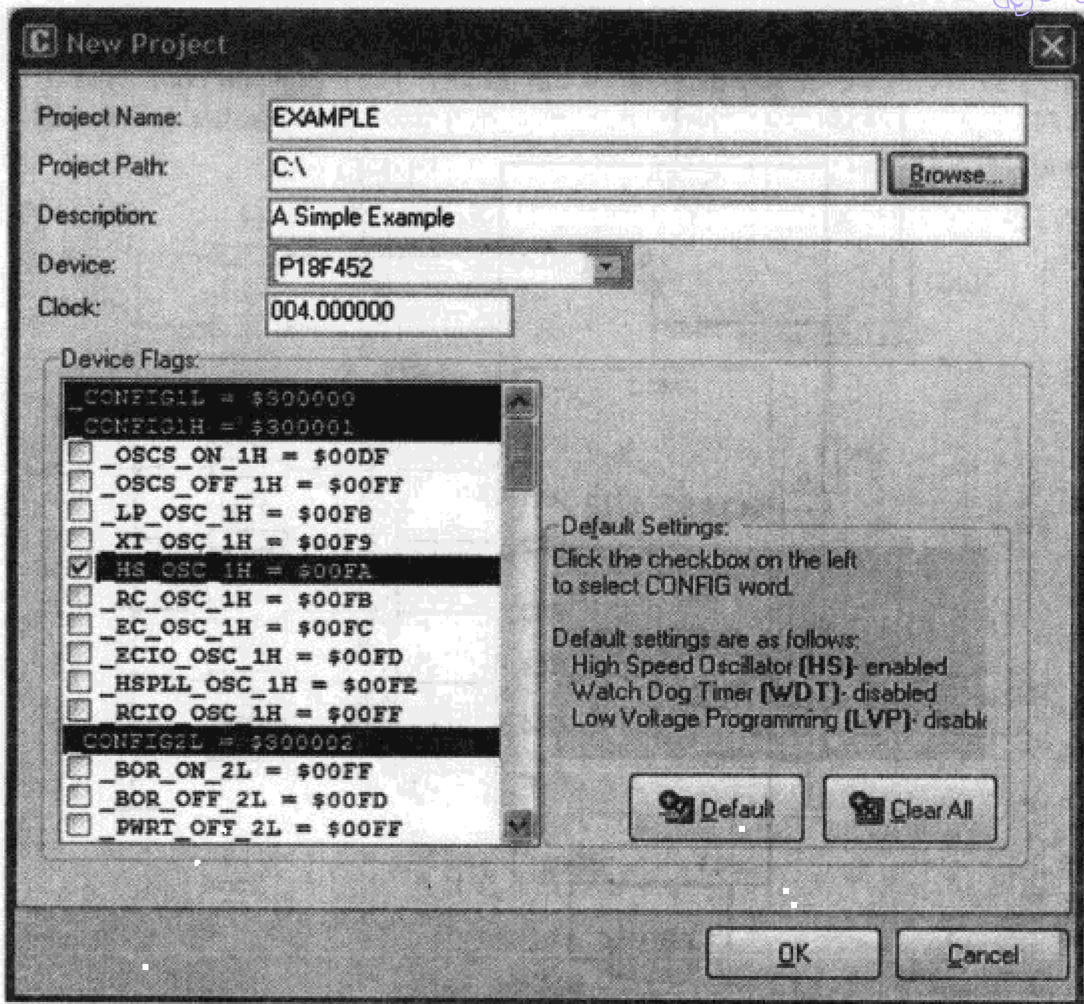


图5-38　创建新项目

步骤4　点击File→save As，将程序保存为EXAMLE。因此，程序将被保存为EXAMPLE.C。

步骤5　按下CTRL+F9或者点击Build Project按钮（如图5-39所示），编译该程序。

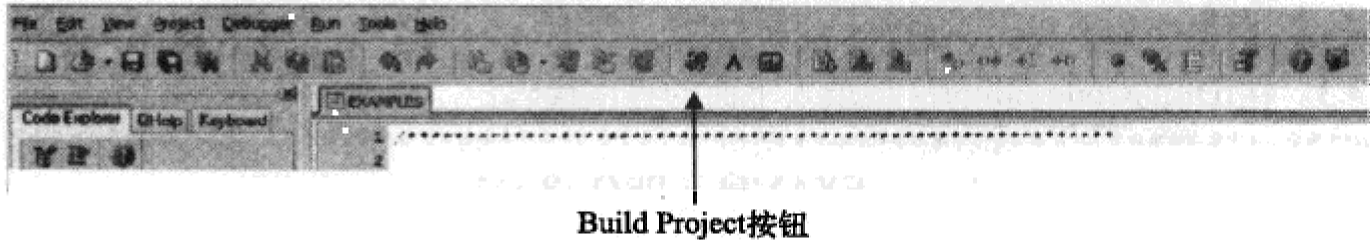


图5-39　Build Project按钮

步骤6　如果编译成功，则在Message窗口中将会出现成功编译的信息，如图5-36所示。程序中出现的任何错误也会显示在Message窗口中，在项目继续进行之前必须得到修正。

编译器将产生大量的输出文件，点击Tool→Options→Output可以对它们进行选择。这些输出文件包括：

261

.ASM文件　这是该程序的汇编语言文件。图5-40给出了EXAMPLE.ASM文件。

```
;  ASM code generated by mikroVirtualMachine for PIC - V. 6.2.1.0
;  Date/Time: 07/07/2007 16:46:12
;  Info: http://www.mikroelektronika.co.yu


; ADDRESS    OPCODE       ASM
; ----------------------------------------------
$0000 $EF04 F000              GOTO   _main
$0008 $        _main:
;EXAMPLE.c,14 ::              void main()
```

图5-40　EXAMPLE.ASM

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

lyw藏书

```
$0008 $6A94                       CLRF  TRISC, 0
;EXAMPLES.c,20 ::                       Sum = 0;
$000A $6A15                       CLRF  main_Sum_L0, 0
$000C $6A16                       CLRF  main_Sum_L0+1, 0
;EXAMPLE.c,21 ::                  for(i=1; i<= 10; i++)
$000E $0E01                       MOVLW 1
$0010 $6E17                       MOVWF main_i_L0, 0
$0012 $0E00                       MOVLW 0
$0014 $6E18                       MOVWF main_i_L0+1, 0
$0016 $       L_main_0:
$0016 $0E00                       MOVLW 0
$0018 $6E00                       MOVWF STACK_0, 0
$001A $5018                       MOVF  main_i_L0+1, 0, 0
$001C $5C00                       SUBWF STACK_0, 0, 0
$001E $E102                       BNZ   L_main_3
$0020 $5017                       MOVF  main_i_L0, 0, 0
$0022 $080A                       SUBLW 10
$0024 $       L_main_3:
$0024 $E307                       BNC   L_main_1
;EXAMPLE.c,23 ::                  SUM = Sum + i;
$0026 $5017                       MOVF  main_i_L0, 0, 0
$0028 $2615                       ADDWF main_Sum_L0, 1, 0
$002A $5018                       MOVF  main_i_L0+1, 0, 0
$002C $2216                       ADDWFC      main_Sum_L0+1, 1, 0
;EXAMPLE.c,24 ::                  }
$002E $       L_main_2:
;EXAMPLE.c,21 ::                  for(i=1; i<= 10; i++)
$002E $4A17                       INFSNZ      main_i_L0, 1, 0
$0030 $2A18                       INCF  main_i_L0+1, 1, 0
;EXAMPLE.c,24 ::                  }
$0032 $D7F1                       BRA   L_main_0
$0034 $       L_main_1:
;EXAMPLE.c,26 ::                  PORTC = Sum;
$0034 $C015 FF82                  MOVFF main_Sum_L0, PORTC
;EXAMPLE.c,27 ::                  }
$0038 $D7FF                       BRA   $
```

图5-40 （续）

.LST文件 这是程序的列表文件。图5-41给出了EXAMPLE.LST文件。 262

```
;  ASM code generated by mikroVirtualMachine for PIC - V. 6.2.1.0
;  Date/Time: 07/07/2007 17:07:12
;  Info: http://www.mikroelektronika.co.yu


; ADDRESS    OPCODE        ASM
; ----------------------------------------------
$0000 $EF04 F000                  GOTO  _main
$0008 $       _main:
;EXAMPLE.c,14 ::                  void main()
;EXAMPLE.c,18 ::                  TRISC = 0;
$0008 $6A94                       CLRF  TRISC, 0
;EXAMPLE.c,20 ::                  Sum = 0;
$000A $6A15                       CLRF  main_Sum_L0, 0
$000C $6A16                       CLRF  main_Sum_L0+1, 0
;EXAMPLE.c,21 ::                  for(i=1; i<= 10; i++)
$000E $0E01                       MOVLW 1
$0010 $6E17                       MOVWF main_i_L0, 0
$0012 $0E00                       MOVLW 0
$0014 $6E18                       MOVWF main_i_L0+1, 0
$0016 $       L_main_0:
$0016 $0E00                       MOVLW 0
$0018 $6E00                       MOVWF STACK_0, 0
$001A $5018                       MOVF  main_i_L0+1, 0, 0
$001C $5C00                       SUBWF STACK_0, 0, 0
$001E $E102                       BNZ   L_main_3
$0020 $5017                       MOVF  main_i_L0, 0, 0
$0022 $080A                       SUBLW 10
$0024 $       L_main_3:
$0024 $E307                       BNC   L_main_1
```

图5-41 EXAMPLE.LST

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
;EXAMPLE.c,23 ::                     SUM = Sum + i;
$0026 $5017                          MOVF   main_i_L0, 0, 0
$0028 $2615                          ADDWF  main_Sum_L0, 1, 0
$002A $5018                          MOVF   main_i_L0+1, 0, 0
$002C $2216                          ADDWFC       main_Sum_L0+1, 1, 0
;EXAMPLE.c,24 ::                     }
$002E $     L_main_2:
;EXAMPLE.c,21 ::                     for(i=1; i<= 10; i++)
$002E $4A17                          INFSNZ       main_i_L0, 1, 0
$0030 $2A18                          INCF   main_i_L0+1, 1, 0
;EXAMPLE.c,24 ::                     }
$0032 $D7F1                          BRA    L_main_0
$0034 $     L_main_1:
;EXAMPLE.c,26 ::                     PORTC = Sum;
$0034 $C015 FF82                     MOVFF main_Sum_L0, PORTC
;EXAMPLE.c,27 ::                     }
$0038 $D7FF                          BRA    $


//** Procedures locations **
//ADDRESS           PROCEDURE
//----------------------------------------------
$0008          main


//** Labels locations **
//ADDRESS           LABEL
//----------------------------------------------

$0008          _main:
$0016          L_main_0:
$0024          L_main_3:
$002E          L_main_2:
$0034          L_main_1:

//** Variables locations **
//ADDRESS           VARIABLE
//----------------------------------------------
$0000          STACK_0
$0001          STACK_1
$0002          STACK_2
$0003          STACK_3
$0004          STACK_4
$0005          STACK_5
$0006          STACK_6
$0007          STACK_7
$0008          STACK_8
$0009          STACK_9
$000A          STACK_10
$000B          STACK_11
$000C          STACK_12
$000D          STACK_13
$000E          STACK_14
$000F          STACK_15
$0010          STACK_16
$0011          STACK_17
$0012          STACK_18
$0013          STACK_19
$0014          STACK_20
$0015          main_Sum_L0
$0017          main_i_L0
$0F82          PORTC
$0F94          TRISC
$0FD8          STATUS
$0FD9          FSR2L
$0FDA          FSR2H
$0FDB          PLUSW2
$0FDC          PREINC2
$0FDD          POSTDEC2
$0FDE          POSTINC2
$0FDF          INDF2
$0FE0          BSR
```

图5-41 （续）

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
$0FE1          FSR1L
$0FE2          FSR1H
$0FE3          PLUSW1
$0FE4          PREINC1
$0FE5          POSTDEC1
$0FE6          POSTINC1
$0FE7          INDF1
$0FE8          WREG
$0FE9          FSR0L
$0FEA          FSR0H
$0FEB          PLUSW0
$0FEC          PREINC0
$0FED          POSTDEC0
$0FEE          POSTINC0
$0FEF          INDF0
$0FF3          PRODL
$0FF4          PRODH
$0FF5          TABLAT
$0FF6          TBLPTRL
$0FF7          TBLPTRH
$0FF8          TBLPTRU
$0FF9          PCL
$0FFA          PCLATH
$0FFB          PCLATU
$0FFD          TOSL
$0FFE          TOSH
$0FFF          TOSU
```

图5-41 （续）

.HEX文件 这是最重要的输出文件，它将会被发送到编程设备以对微控制器进行编程。图5-42给出了EXAMPLE.HEX文件。

```
:1000000004EF00F0FFFFFFFF946A156A166A010E05
:10001000176E000E186E000E006E1850005C02E1A4
:1000200017500A0807E3175015261850162 2174ACA
:10003000182AF1D715C082FFFFD7FFFFFFFFFFFF90
:020000040030CA
:0E000000FFF9FFFEFFFFFBFFFFFFFFFFFFFF0B
:00000001FF
```

图5-42 EXAMPLE.HEX

5.3.3 仿真器的使用

使用软件仿真器（release模式），按照以下的步骤，可以对5.3.2节的程序进行仿真。软件仿真不需要硬件的参与。

例5.2 描述例5.1中所开发程序的仿真步骤。在仿真期间，单步运行程序，显示不同变量的值和PORTC的状态。请指出PORTC上最后显示的值。

解 程序仿真所需的步骤如下。

步骤1 启动mikroC IDE，确认例5.1的程序显示在Code Editor区域中。

步骤2 从下拉菜单中选择Debugger→Select Debugger→Software PIC Simulator，如图5-43所示。

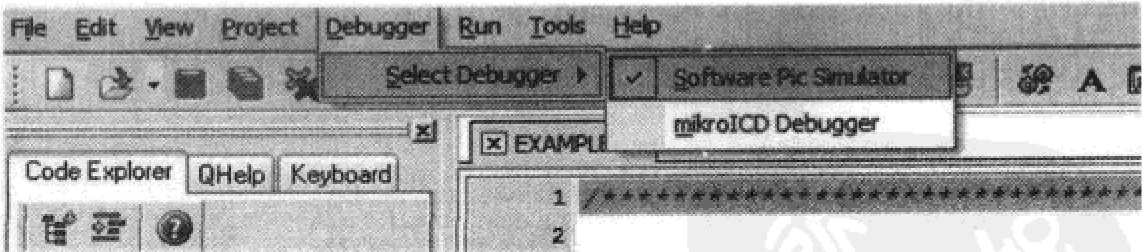


图5-43 选择调试器

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

263
～
265

步骤3 从下拉菜单中选择Run→Start Debugger。将看到调试器窗口，如图5-44所示。

步骤4 选择需要在仿真过程中观察的变量。假设需要显示变量Sum、i和PORTC的值：

- 点击Select from variable list按钮，然后找到并点击变量名Sum；
- 点击Add按钮，将这个变量添加到Watch list；
- 重复以上步骤，添加变量i和PORTC。

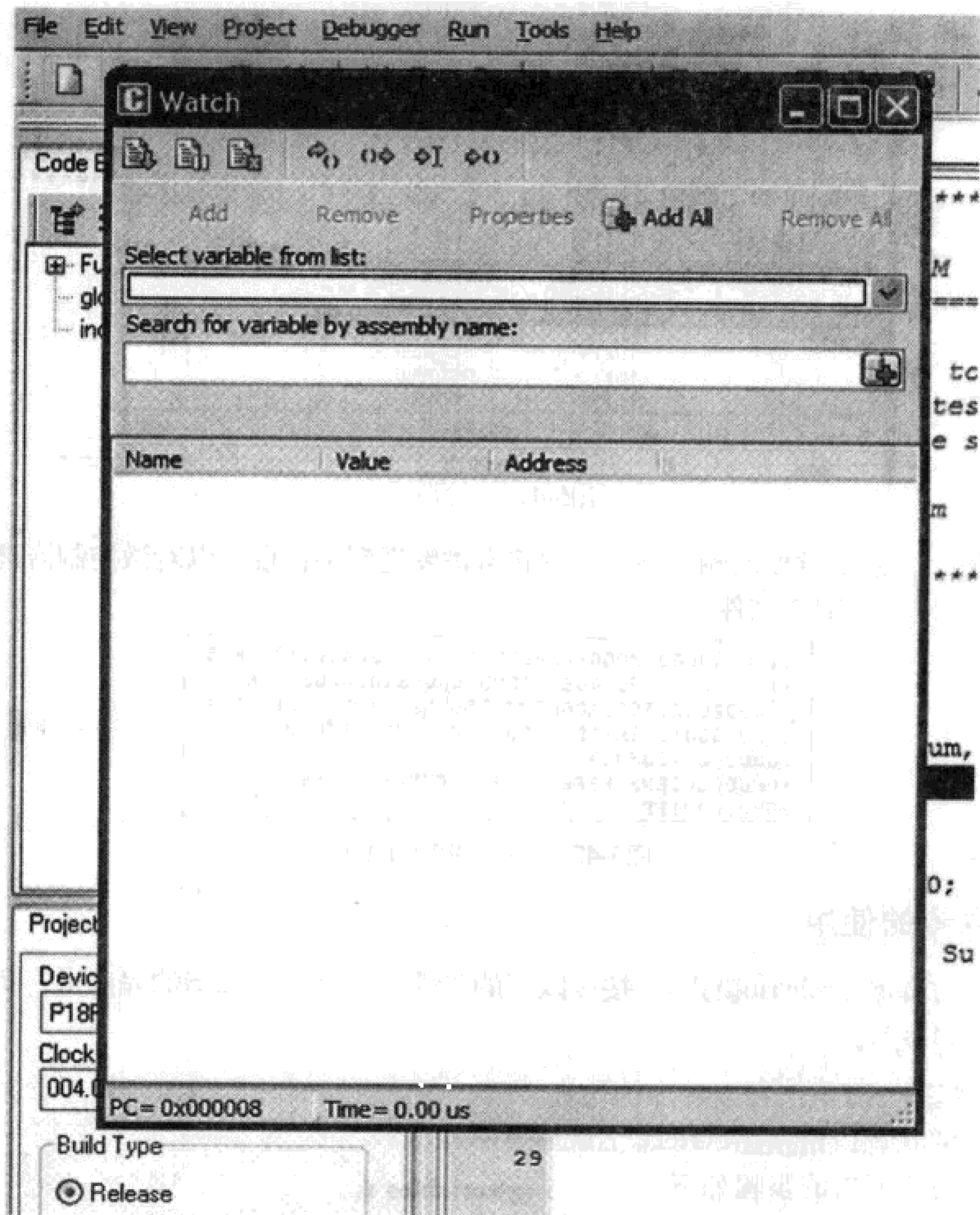


图5-44 启动调试器

此时，调试器窗口就会如图5-45所示。

步骤5 现在，可以单步运行程序，并观察变量的变化。

在键盘上按下F8键。读者可以看到一个蓝色的行在向下移动。这是程序当前执行到的行。连续按F8直到进入循环，读者就可以看到变量Sum和i已经变为“1”了，如图5-46所示。最近发生过变化的项将呈现红色。在Watch窗口中双击任何一项，都可以打开Edit Value窗口，在这里可以改变变量的值和寄存器的内容，也可以将数据换个方式显示，如十进制、十六进制、二进制，或者是浮点数和字符。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

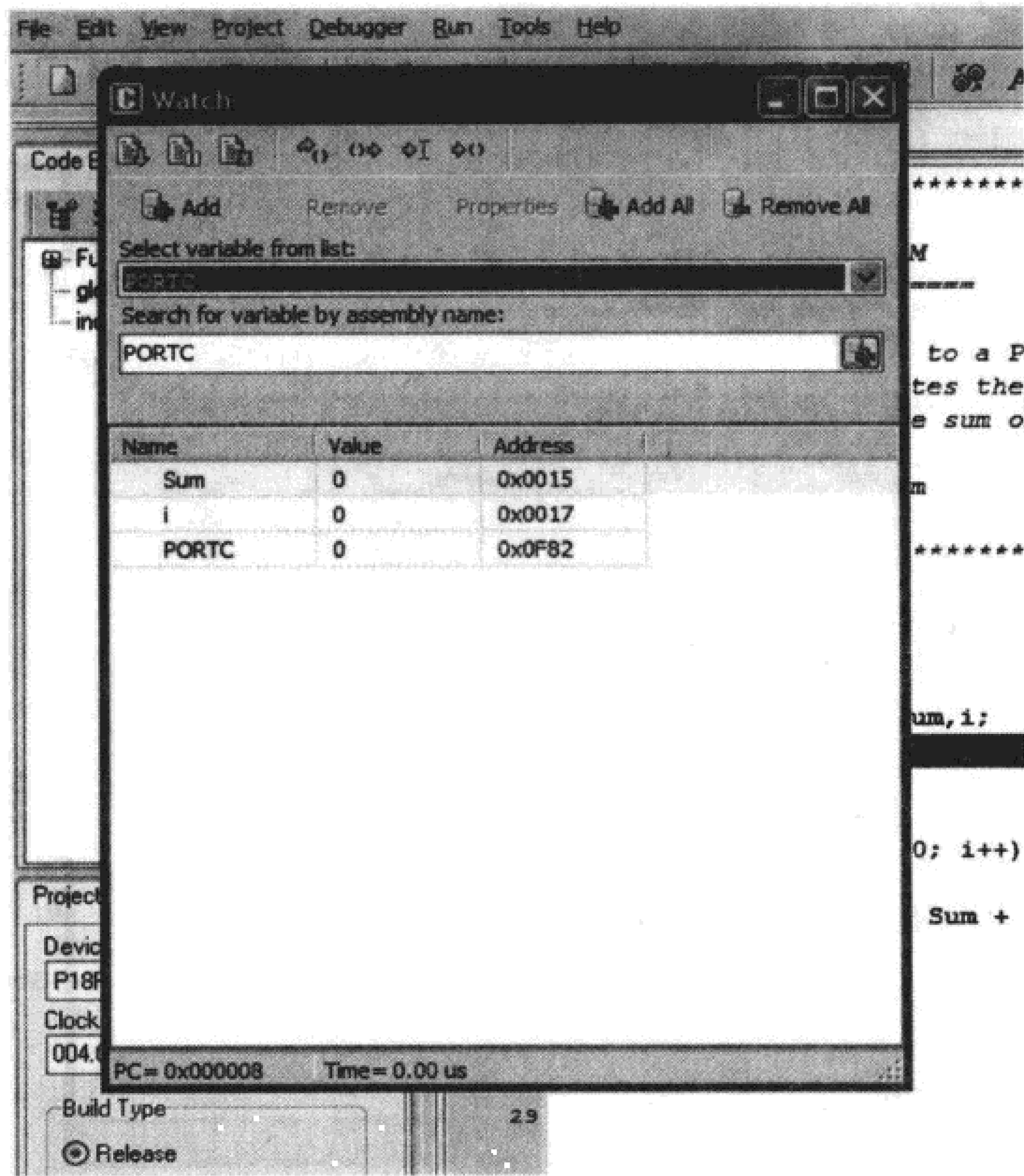


图5-45 选择需要显示的变量

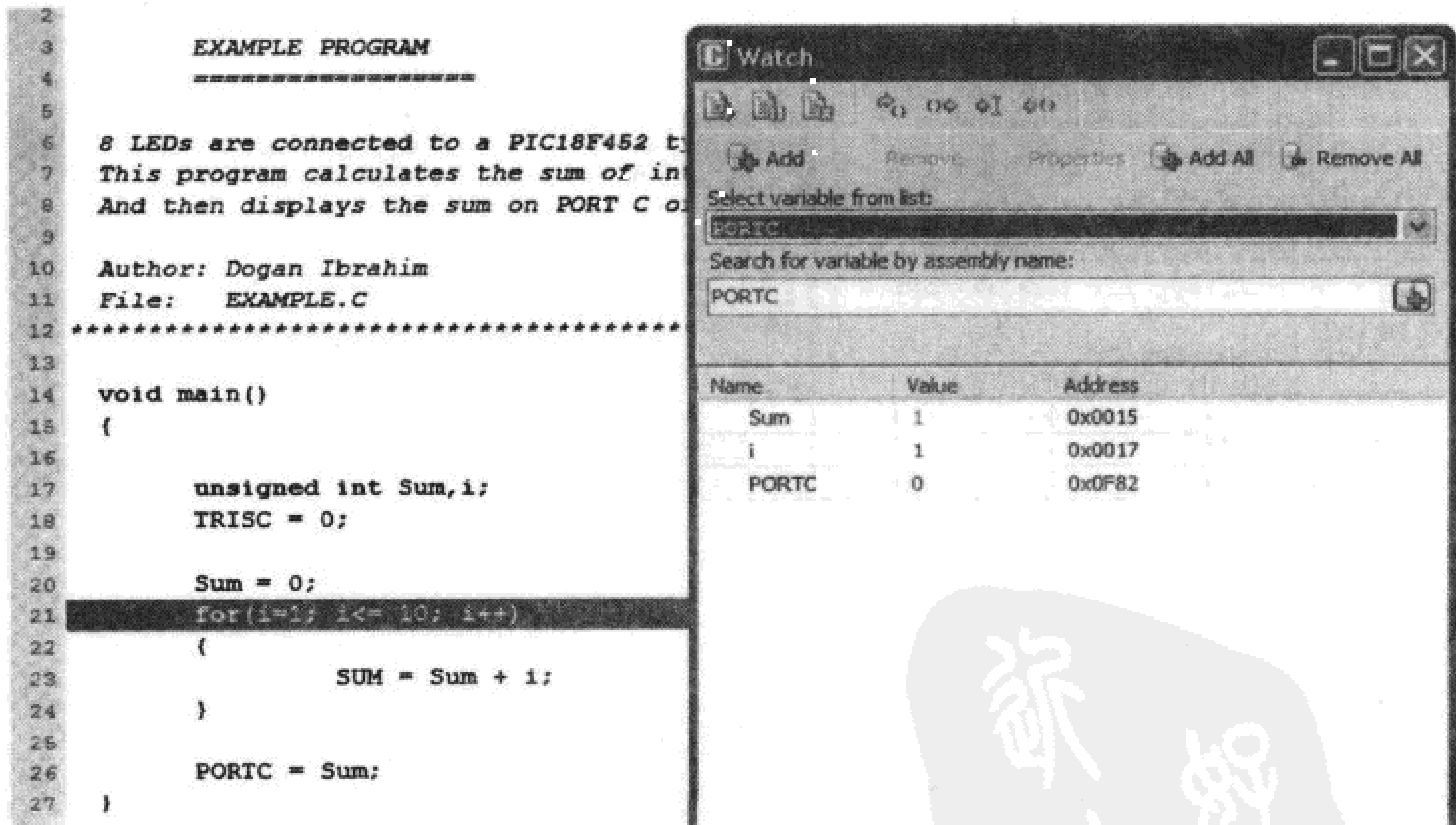


图5-46 单步运行程序

世纪电源网·论坛
BBS.21dianyuan.com
电源工程师
设计灵感之源

tyw藏书

步骤6　连续按F8键，直到程序跳出for循环，执行将数据发送到PORTC的程序行。如图5-47所示，此时i=11，Sum=55。

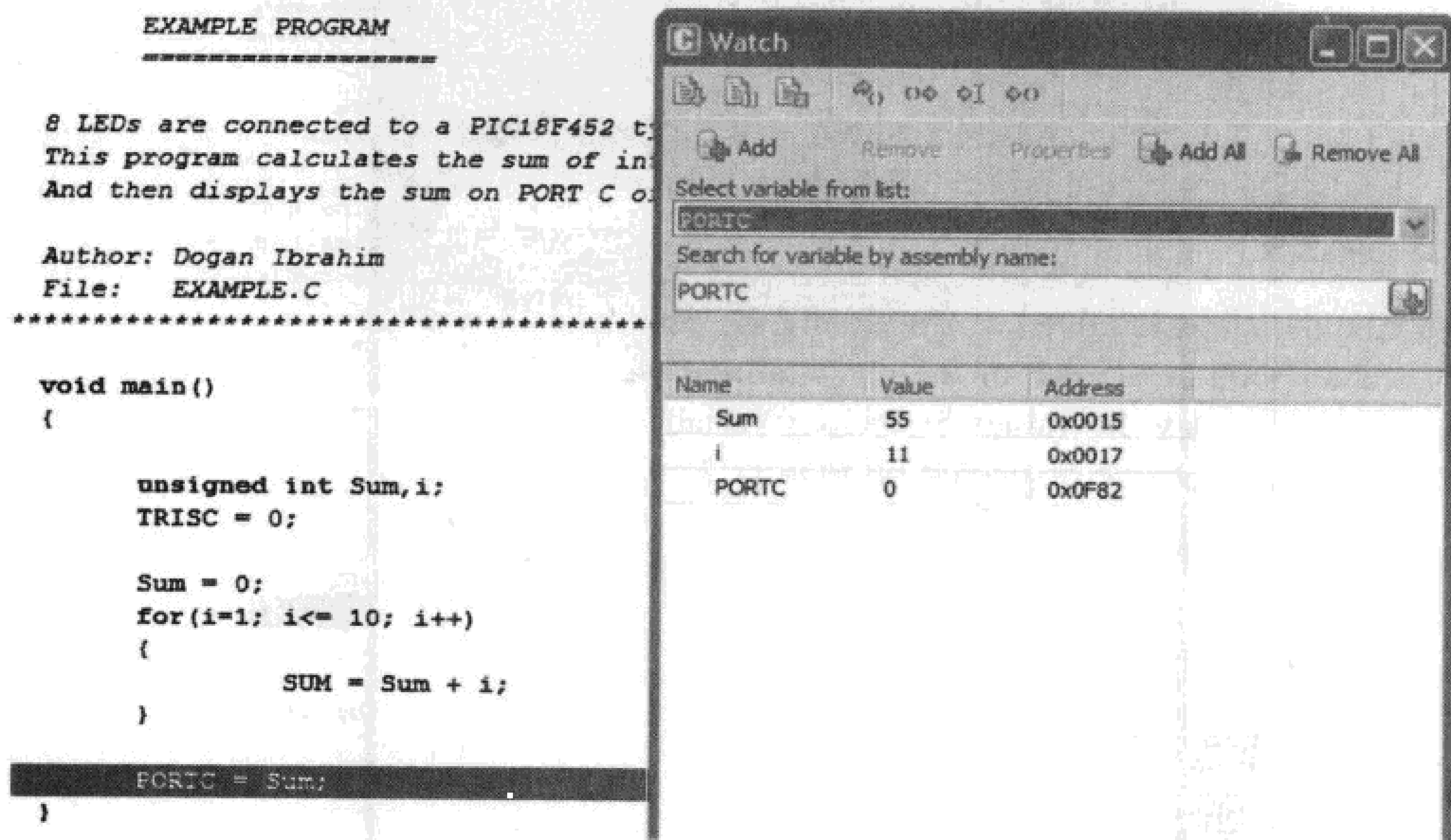


图5-47　单步运行程序

步骤7　再次按下F8键，向PORTC发送变量Sum的值。如图5-48所示，此时PORTC将会收到数值55，它就是1到10各数的总和。

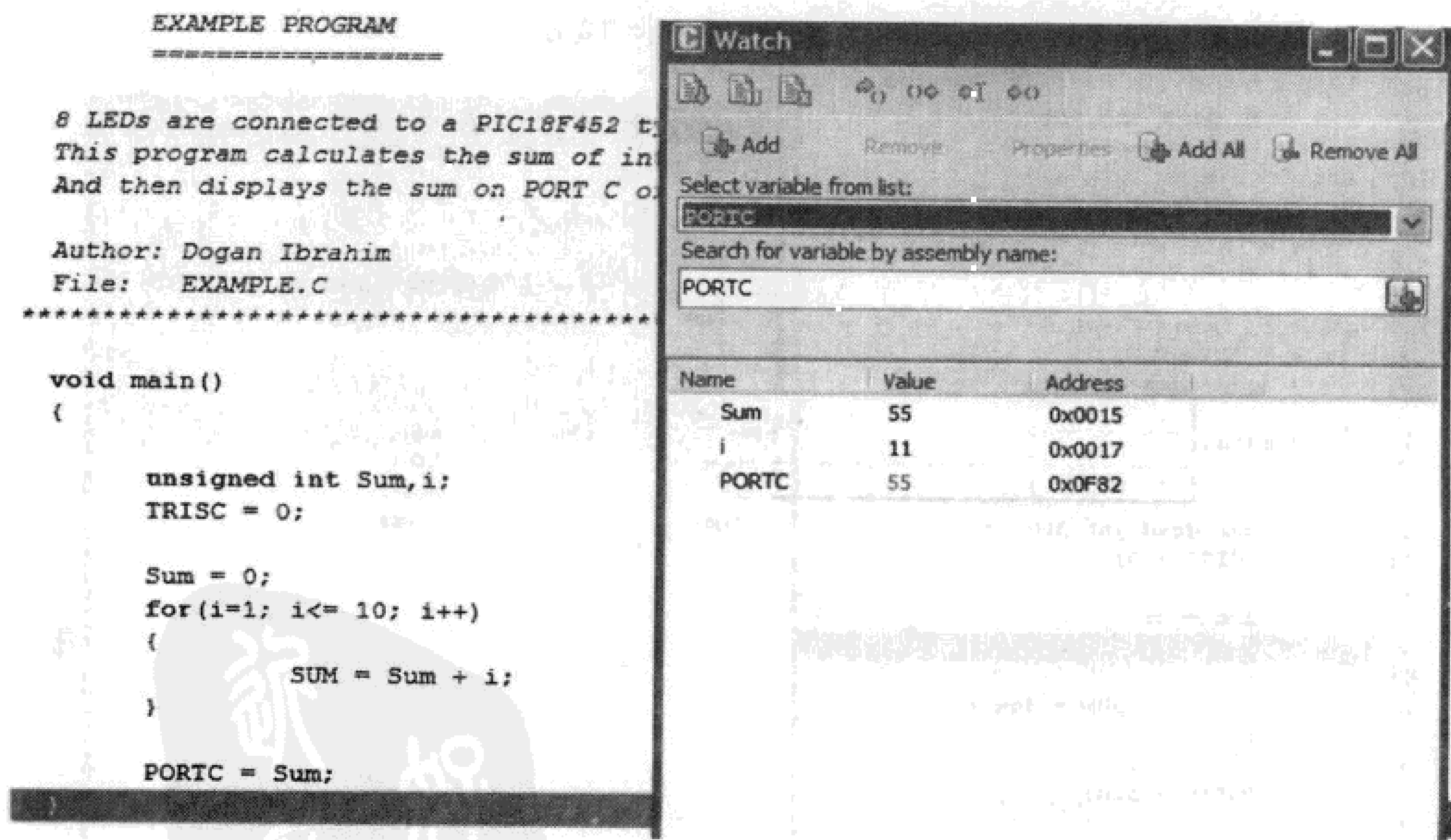


图5-48　PORTC得到数值55

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源

tyw藏书

到此，仿真结束。从下拉菜单中选择Run→Stop Debugger。

在上述的仿真例子里，从头到尾单步运行程序，查看到了PORTC显示的值。下面的例子将说明如何在程序中设置断点，并运行程序，直到遇到断点。

例5.3 描述例5.1里所开发程序的仿真步骤。在程序的结尾处设置一个断点，让调试器运行至该断点处。在该断点处，显示各个变量的值和PORTC状态。请指出PORTC最后显示的状态。

解 程序仿真所需的步骤如下。

266 ~ 270

步骤1 启动mikroC IDE，确认例5.1的程序出现在Code Editor窗口中。

步骤2 从下拉菜单中选择Debugger→Select Debugger→Software PIC Simulator。

步骤3 从下拉菜单中选择Run→Start Debugger。

步骤4 在Watch窗口中选择变量Sum、i和PORTC，如例5.2所示。

步骤5 在程序结尾处设置一个断点。在程序最后的闭括号处（即第27行）点击鼠标，然后按下F5键。如图5-49所示，读者可以发现断点处变为红色，Code Editor窗口的左边栏目多了一个小标记。

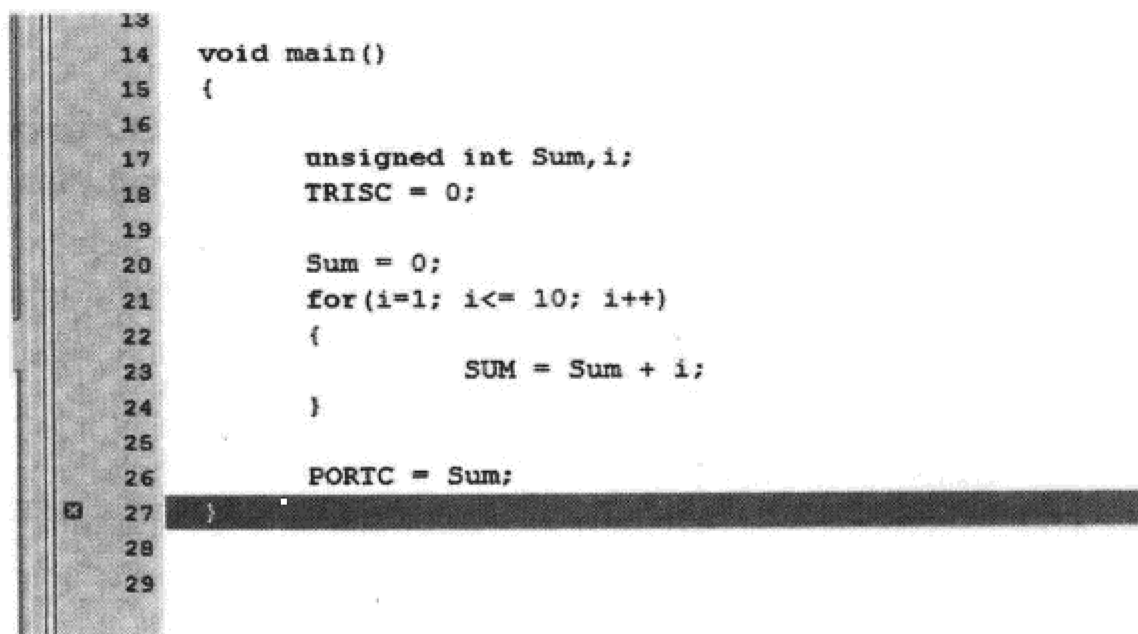


图5-49 在第27行设置断点

步骤6 现在打开调试器，按下F6键，运行程序。程序停在断点处，并显示出当前变量，如图5-48所示。

到此，仿真结束。从下拉菜单中选择Run→Stop Debugger。

为了清除断点，只需把光标移动到断点所在行，然后再按F5键。为了清除所有设置的断点，可以按SHIFT+CTRL+F5组合键。为了显示程序中设置的断点，可以按SHIFT+F4组合键。

下面给出另外一些有用的调试命令。

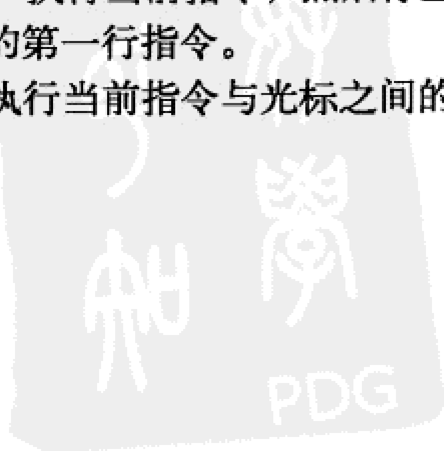
271

Step Into[F7] 执行当前指令，然后停止。如果当前指令是调用一个子程序，那么程序将会进入被调用的子程序，并停止在第一行指令。

Step Over[F8] 执行当前指令，然后停止。如果当前指令是调用一个子程序，那么程序将跳过子程序，并停止在调用程序后面的第一行指令。

Step Out[CTRL+F8] 执行当前指令，然后停止。如果指令位于子程序内，则执行指令，然后停止在调用程序后面的第一行指令。

Run to Cursor[F4] 执行当前指令与光标之间的所有指令。



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

Jump to Interrupt[F12] 跳到中断服务程序入口地址（PIC18微控制器的中断程序入口地址为0x08），并执行该地址处的程序。

5.3.4 mikroICD 内电路调试器的使用

本节将讨论如何使用mikroICD内电路调试器（也称作PICFlash 2编程器）来调试例5.1中开发的程序。首先，必须建立起硬件系统，并连接内电路调试设备。在这个例子里，硬件系统是搭建在面包板上的，使用PICFlash 2 mikroICD内电路调试器来调试系统。注意，引脚RB6和RB7是被mikroICD使用的，当mikroICD处于激活状态时，它们不作I/O端口使用。

1. 电路图

该项目的电路图如图5-50所示。mikroICD内电路调试器通过微控制器的下列引脚连接到开发电路系统：

- MCLR
- RB6
- RB7
- +5 V
- GND

272



图5-50 项目电路图

mikroICD有两种工作模式。在被动模式下，调试器用到的所有微控制器引脚都要连接到开发系统。在主动模式下，微控制器的引脚MCLR、RB6和RB7不与开发系统相连，而是用来对微控制器进行编程。在编程完成后，可以再接回系统。

mikroICD调试设备有一个10线的IDC连接器，可以通过10线的接头与目标系统相连。当开发完成且mikroICD调试器拆除后，可将两个IDC接头通过跳线相连。图5-51展示了在面包板上搭建的电路系统。

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源

tyw藏书

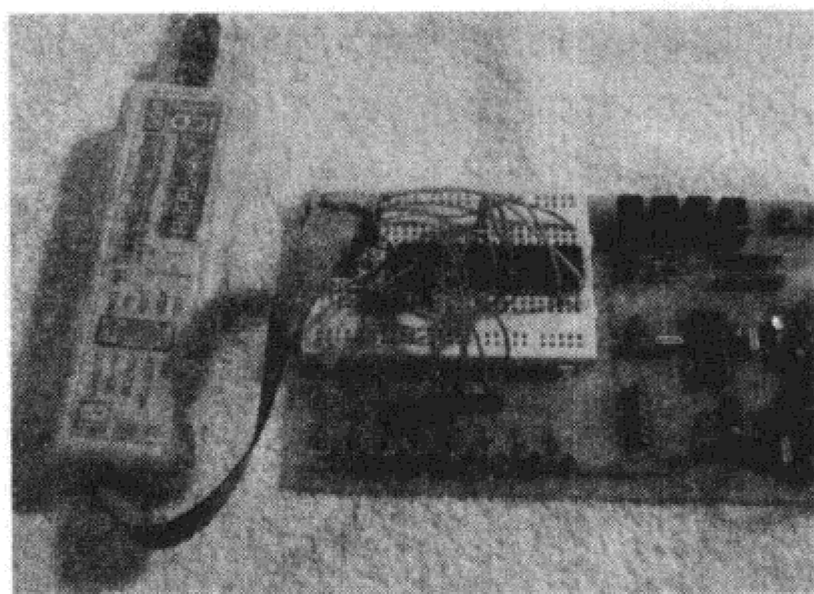


图5-51 搭建在面包板上的电路系统

2. 调试

在建立好硬件系统后，可以对微控制器进行编程，并使用内电路调试器对系统的工作进行测试。其步骤如下。

273

步骤1 打开mikroC IDE，确定例5.1开发的程序显示在Code Editor窗口中。

步骤2 点击Edit Project（编辑工程）按钮（如图5-52所示），设置DEBUG_ON，如图5-53所示。

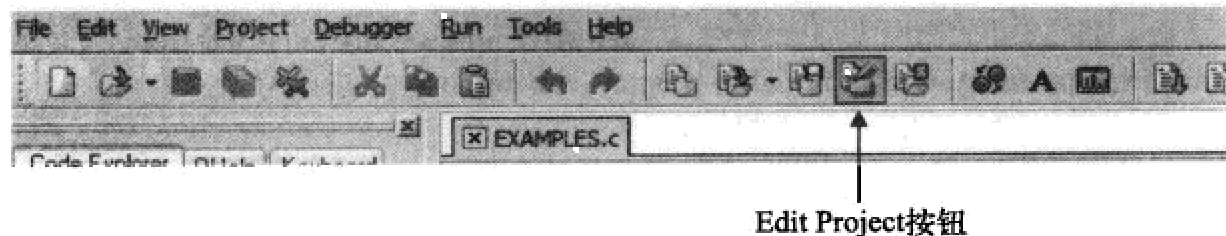


图5-52 Edit Project按钮

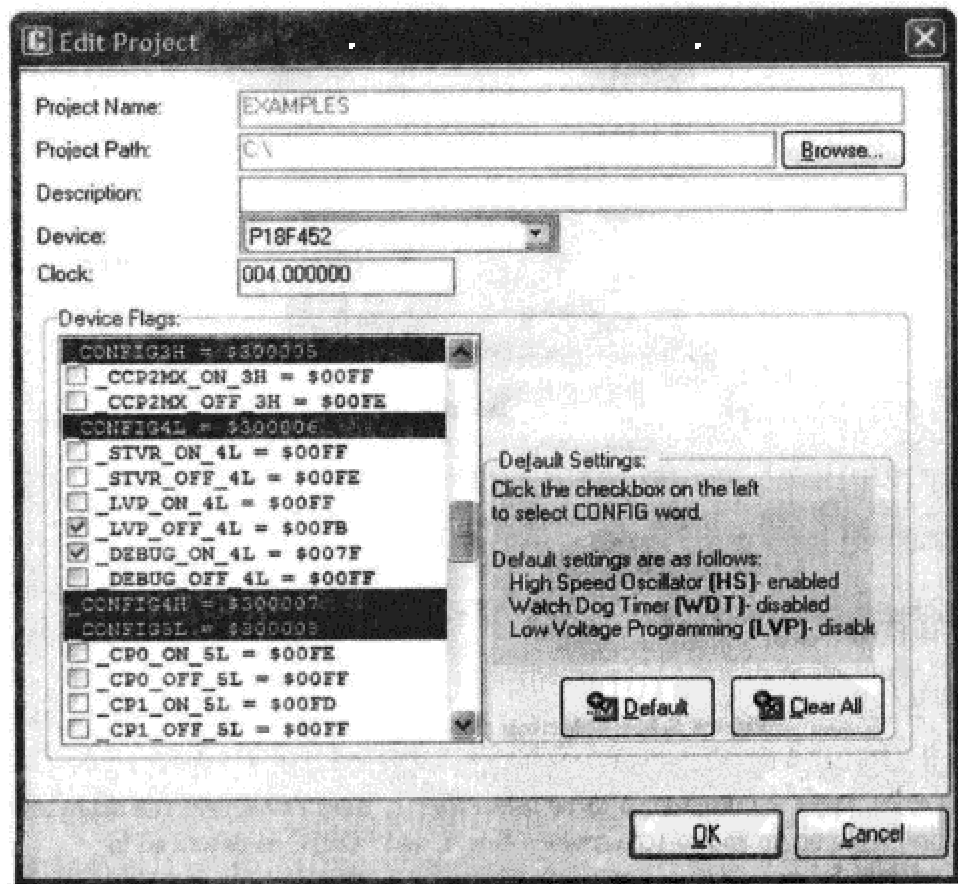
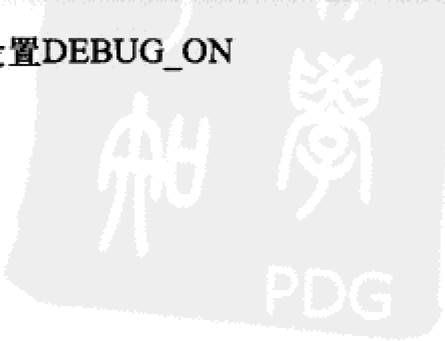


图5-53 设置DEBUG_ON



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

步骤3　在Project Setup窗口中选择ICD Debug，如图5-54所示。

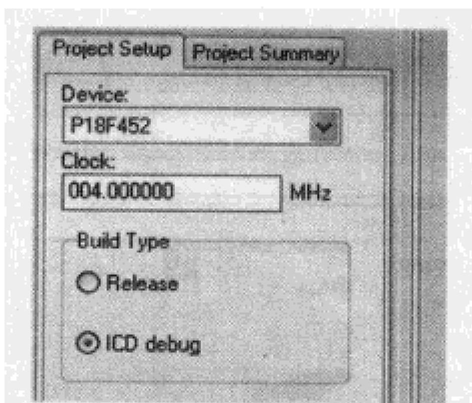


图5-54　选择ICD Debug

步骤4　点击Build Project图标，使用调试器开始编译程序。在编译成功后，将在Message

274

窗口中看到信息“Success (ICD Build)”。

步骤5　确认mikroICD调试设备已正确连接，如图5-50所示。从下拉菜单中选择Tools→PicFlash Programmer，对微控制器进行编程。

步骤6　从下拉菜单中选择Debugger→Select Debugger→mikroICD Debugger，如图5-55所

275

示。

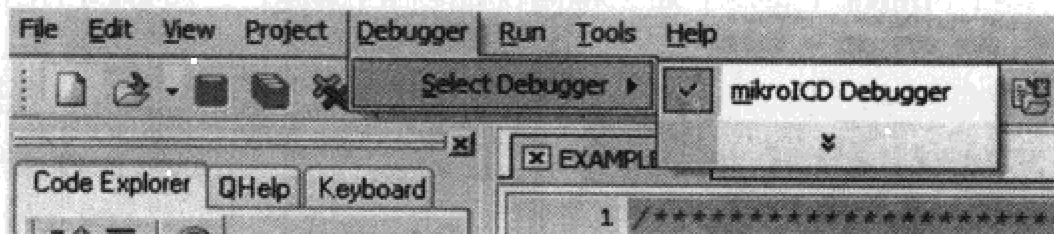
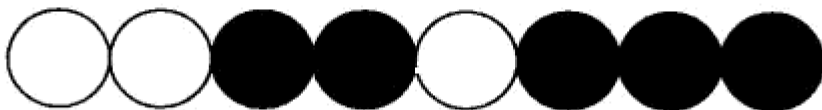


图5-55　选择mikroICD调试器

步骤7　从下拉菜单中选择Run→Start Debugger。将弹出调试器表单，选择变量Sum、i和PORTC，如例5.2所示。

步骤8　按F8键，单步运行程序，将会看到变量值的变化情况。在程序的最后，十进制数55将被发送到PORTC，第0、1、2、4和5位的LED将被点亮，如图5-56所示。



276

图5-56　十进制数55的LED显示

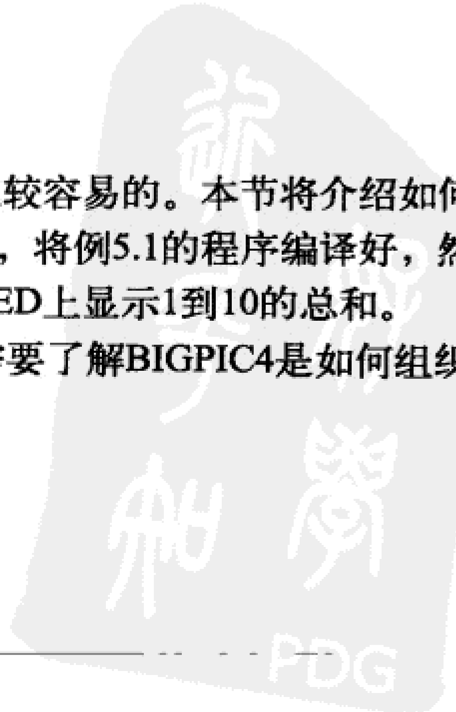
步骤9　停止调试。

在包含有延迟的子程序中，使用Step Into[F7]和Step Over[F8]是相当耗时的。在这种情况下，应该使用Run to Cursor[F4]和断点代替。

5.3.5　开发板的使用

使用开发板来开发基于微控制器的项目是比较容易的。本节将介绍如何使用前面提及的开发板BIGPIC4。使用板上mikroICD内电路模拟器，将例5.1的程序编译好，然后下载到微控制器中。接下来，运行程序，并在连接到PORTC的LED上显示1到10的总和。

尽管如此简单，在使用开发板之前，还是需要了解BIGPIC4是如何组织以及如何在板上使用各种设备的。



世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

BIGPIC4开发板

图5-57给出了BIGPIC4开发板的视图，并用箭头标出了各部件的功能。开发板既可以使用外部电源供电（8 V～16 V的AC/DC适配器），也可以使用PC的USB端口供电，具体选择可通过跳线设置。在这个应用中，开发板从USB端口获取工作电源。

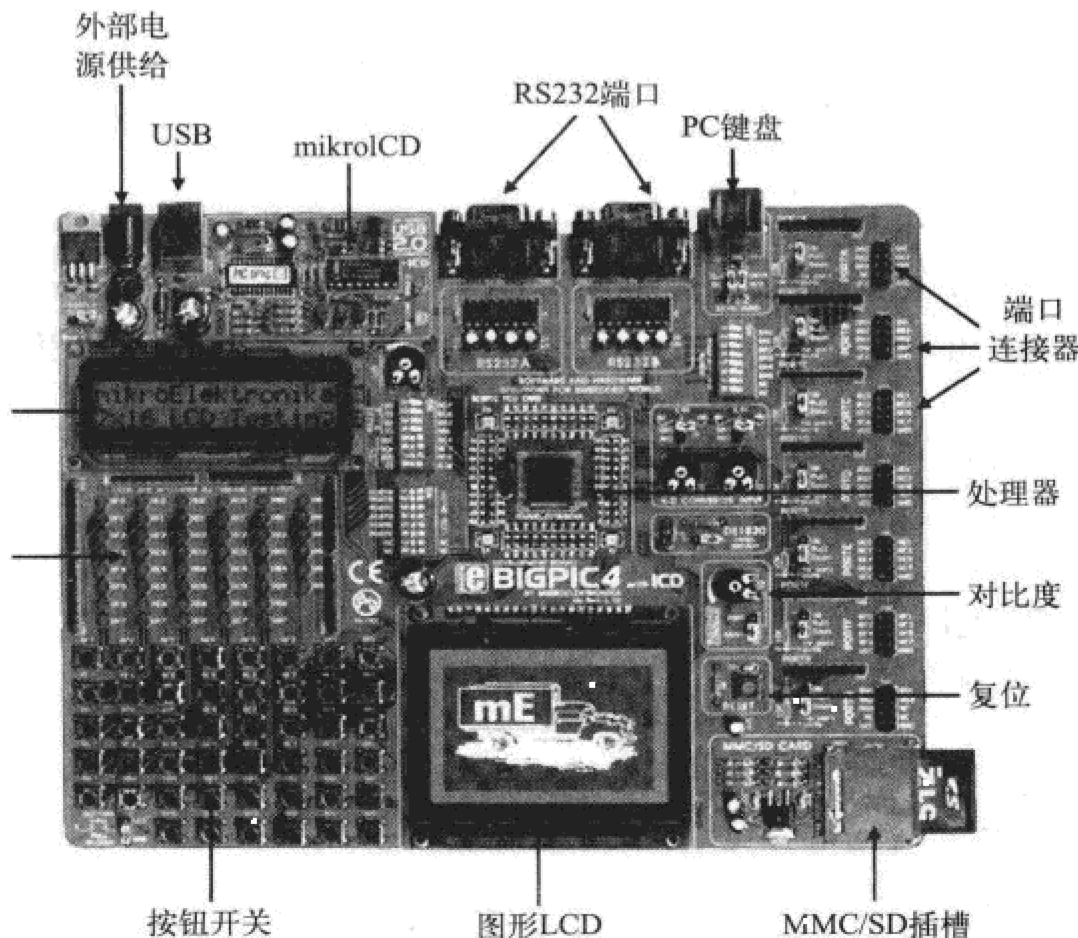


图5-57 BIGPIC4开发板

在开发板的左上角，连接有一个2行16列的LCD显示器。LCD的对比度可以通过一个小型电位计来调节。

开发板上的46个LED可以连接到微控制器的输出端口，这可用开关S2来选择。图5-58以PORTC为例说明了LED的选择。将1kΩ的电阻和LED串联，用来限制电流。例如，要连接8个LED到PORTC，必须将标有PORTC的开关S2拨至ON的位置。

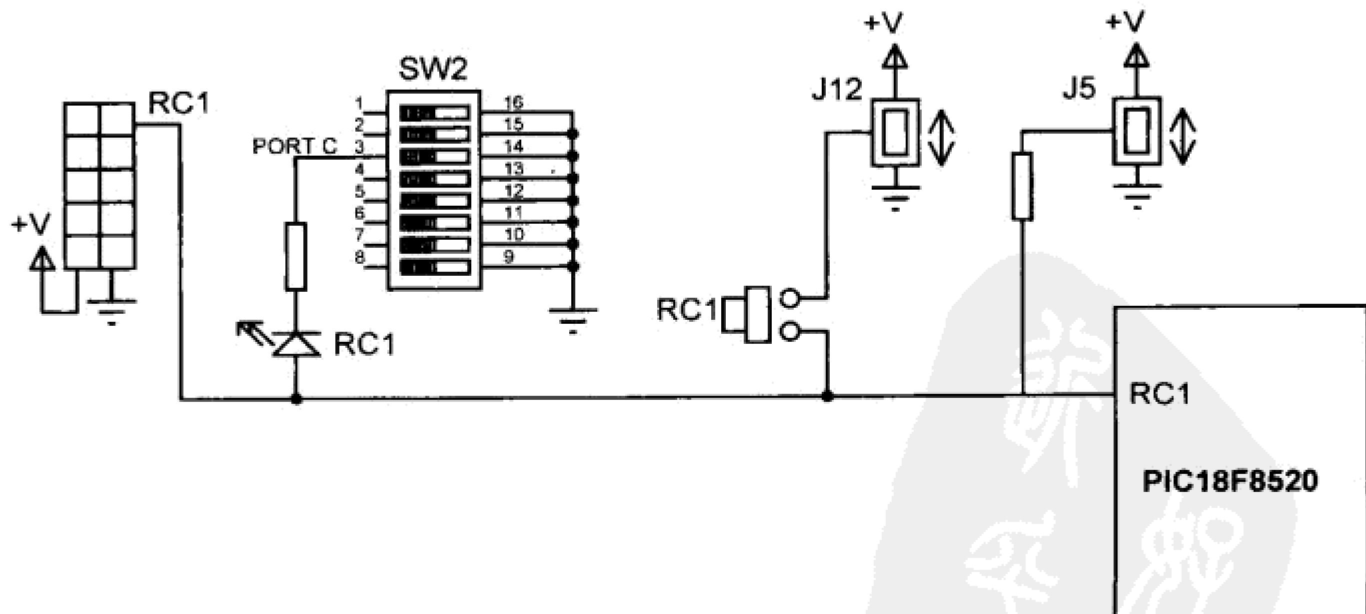


图5-58 LED和按钮开关的连接

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

277~278

开发板上的46个按钮开关也可以用来编程微控制器的数字输入。此外，还有一个按钮开关可用作复位键RESET。跳线J12用来决定当按下一个按钮时，向微控制器输入的是产生逻辑0还是逻辑1。如果按键没有被按下，则引脚的状态由跳线J5决定。

在底部中心区域，有一个128×64像素的图形LCD连接到开发板。LCD的对比度可以通过一个小型电位计来调节。

在开发板的右下角是MMC/SD卡插槽，可以支持2GB的存储容量。

复位键RESET在MMC/SD卡插槽的上方。

在复位键的上面，有两个用于模数转换器应用的电位计。

在开发板的右边，微控制器的所有端口在连接器处都是可用的。而在开发板的正上方，有两个RS232端口和一个PC键盘的接口。

开发板支持64脚和80脚的微控制器。开发板已带有一块PIC18F8520微控制器芯片，晶振频率为10MHz。

关于开发板的更多详情，请参阅BIGPIC4用户手册。

使用BIGPIC4开发板开发项目的步骤如下。

步骤1　双击mikroC图标，打开IDE。

步骤2　创建一个新项目EXAMPLE2（如图5-59所示），选择微控制器的型号为PIC18F8520，时钟频率为10MHz，设备标志为：

- _OSC_HS_1H
- _WDT_OFF_2H
- _LVP_OFF_4L
- _DEBUG_ON_4L

279

步骤3　在IDE的Code Editor窗口中录入以下程序代码：

```
/*****************************************************************
                         EXAMPLE PROGRAM

This program uses the PICBIG4 Development Board. 8 LEDs are connected
To PORTC of the microcontroller which is a PIC18F8520 operating at 10MHz.
This program calculates the sum of integer numbers from 1 to 10
And then displays the sum on PORTC of the microcontroller.
Author:   Dogan Ibrahim
File:     EXAMPLE2.C
*****************************************************************/
void main()
{
     unsigned int Sum,i;
     TRISC=0;

     Sum=0;
     for(i=i ; i<=10;i++)
     {
               Sum=Sum + i;
     }
     PORTC = Sum;
}
```

步骤4　点击File→Save As，把程序保存为EXAMPLE2。

步骤5　在Project Setup窗口中选择ICD Debug。按CTRL+F9组合键或者点击Build Project按

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

钮，编译程序。

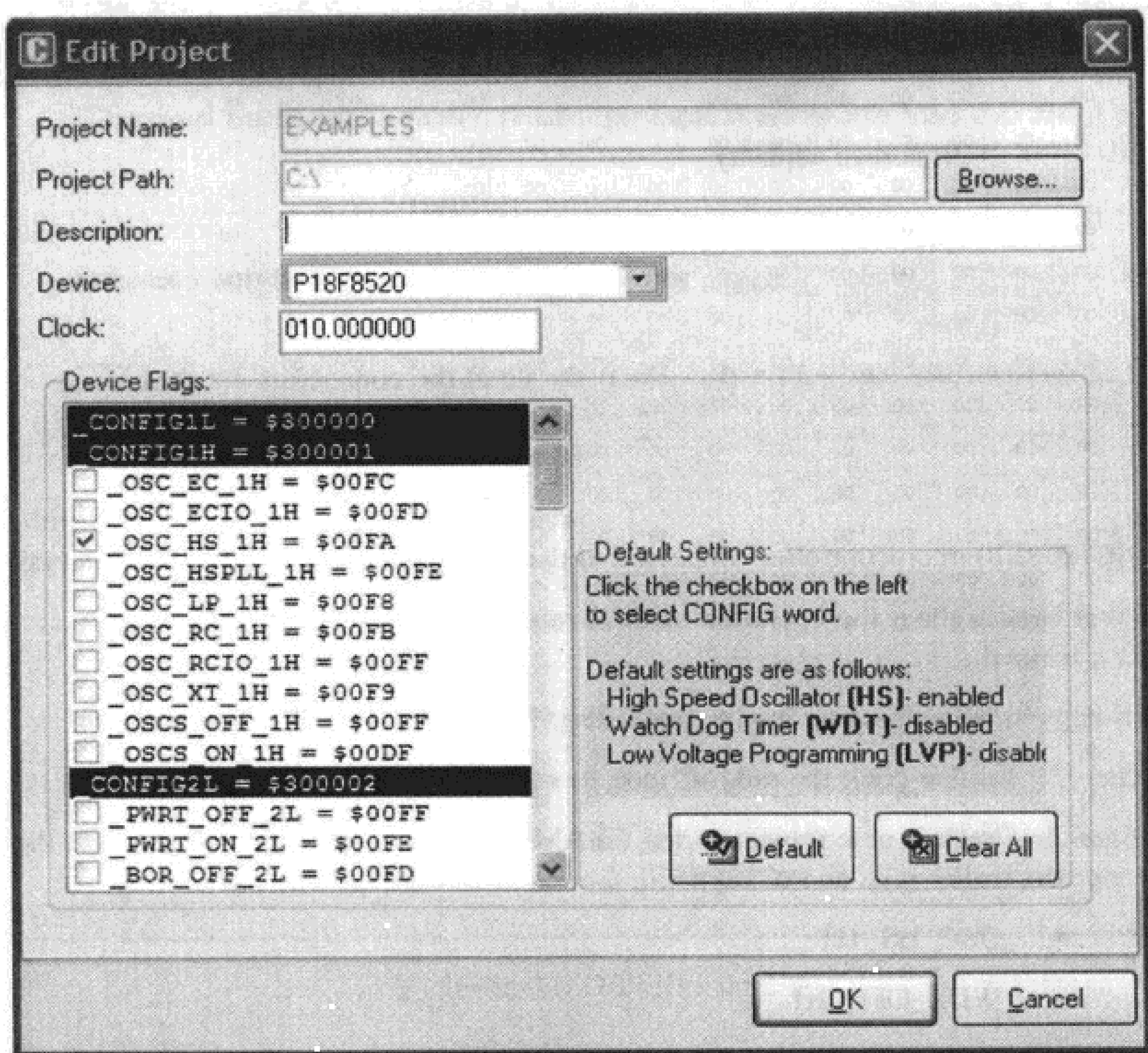


图5-59 创建新项目

步骤6 将BIGPIC4开发板连接到计算机的USB端口。配置LED和PORTC的连接，将标有PORTC的开关S2拨到ON位置。

步骤7 从下拉菜单中选择Tools→PicFlash Programmer，对微控制器进行编程。

步骤8 选择Debugger→Select Debugger→mikroICD Debugger。

步骤9 点击Run→Start Debugger，打开调试器，并从Watch窗口中选择变量Sum、i和PORTC。

步骤10 重复地按F8键，单步运行程序直到结束。在程序的末尾，PORTC的LED将会被点亮，显示出十进制数55（第0、1、2、4、5位的LED被点亮）。

步骤11 结束调试。

查看EEPROM窗口 当选择好mikroICD调试模式并开始调试时，可从mikroC IDE下拉菜单中激活mikroICD EEPROM窗口，它将显示PIC内部EEPROM存储器的内容。为了观察存储器，点击View→Debug Windows→View EEPROM。图5-60给出了EEPROM窗口的示例。 281

查看RAM窗口 当选择好mikroICD调试模式并开始调试时，可从mikroC IDE下拉菜单中激活mikroICD RAM窗口，它将显示PIC内部RAM的内容。为了查看RAM，点击View→Debug Windows→View RAM。图5-61给出了RAM窗口的示例。

查看代码窗口 当选择好mikroICD调试模式并开始调试时，可从mikroC IDE下拉菜单中激活mikroICD Code窗口，它显示PIC内部的代码内存。为了查看代码内存，点击View→Debug Windows→View Code。图5-62给出了Code窗口的示例。 282

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

EEPROM

Write Eeprom Read EEprom

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
0000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

Status: Idle

图5-60 EEPROM存储器的显示

RAM

RAM History

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
0000	A0	16	24	10	10	4C	1C	11	80	00	B7	91	00	00	30	10	...
0010	81	1B	04	C0	00	51	20	82	50	C0	68	06	51	28	48	0A	...
0020	18	05	01	32	CC	C3	04	01	12	00	62	02	02	44	04	82	...
0030	02	18	88	A2	60	88	40	A2	00	5A	80	C0	41	C9	00	20	...
0040	32	3A	84	FE	C2	40	11	8F	15	0C	01	C2	08	06	08	06	...
0050	1A	01	40	04	00	82	00	CC	17	B4	10	C0	52	03	03	61	...
0060	46	A4	42	60	1B	66	28	05	4A	60	4C	50	02	02	13	37	...
0070	00	76	74	46	50	81	00	07	35	99	20	6A	20	03	99	E0	...
0080	C9	02	C8	02	2A	CD	02	12	0C	A1	A4	12	10	19	55	14	...
0090	88	D6	02	10	D0	02	40	53	9D	00	40	29	A0	08	40	08	...
00A0	36	29	01	09	02	0E	01	40	01	20	12	13	2B	0C	38	E2	...
00B0	60	51	74	04	62	30	34	32	04	C1	90	BD	82	C0	C8	A4	...
00C0	48	32	0C	29	0D	92	00	88	00	FA	CB	86	09	A1	63	90	...
00D0	20	09	81	81	D7	08	0A	10	00	43	44	84	55	02	01	0C	...
00E0	21	04	21	08	04	85	00	4D	99	04	88	11	04	10	00	0C	...
00F0	44	A3	46	45	00	21	28	92	0E	00	C7	30	18	0E	20	2B	...

图5-61 RAM存储器的显示

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com
lyw藏书

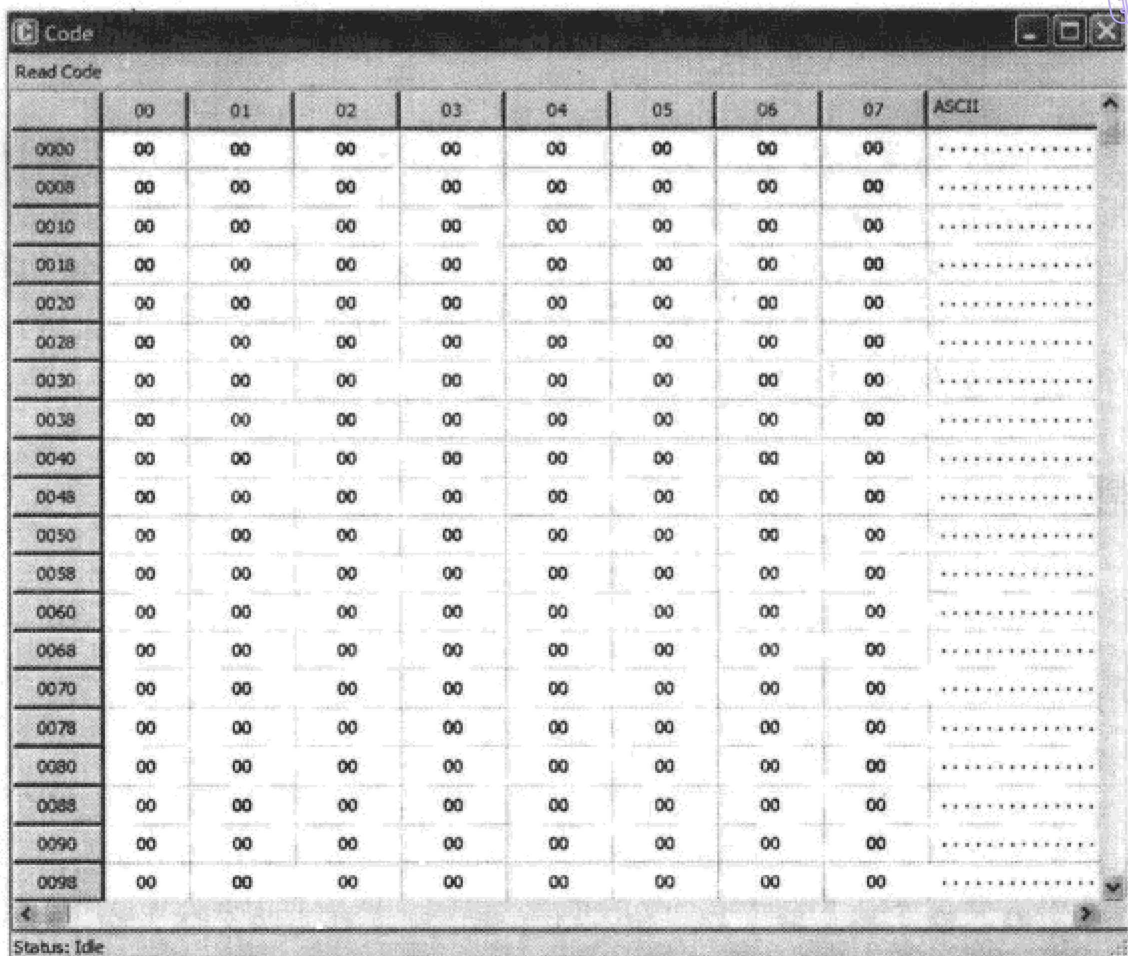


图5-62 代码内存的显示

查看统计数据 可从mikroC IDE下拉菜单中选择Statistics窗口，它显示关于程序的各种统计数据。为了查看统计数据窗口，点击View→View Statistics。图5-63给出了统计数据窗口的示例，它包含了几个标签。Memory Usage标签显示了所用的RAM（数据存储器）和ROM（代码存储器）的容量。Procedures标签显示了进程的数量和位置。RAM和ROM标签显示了存储器的详细内容。

283~284

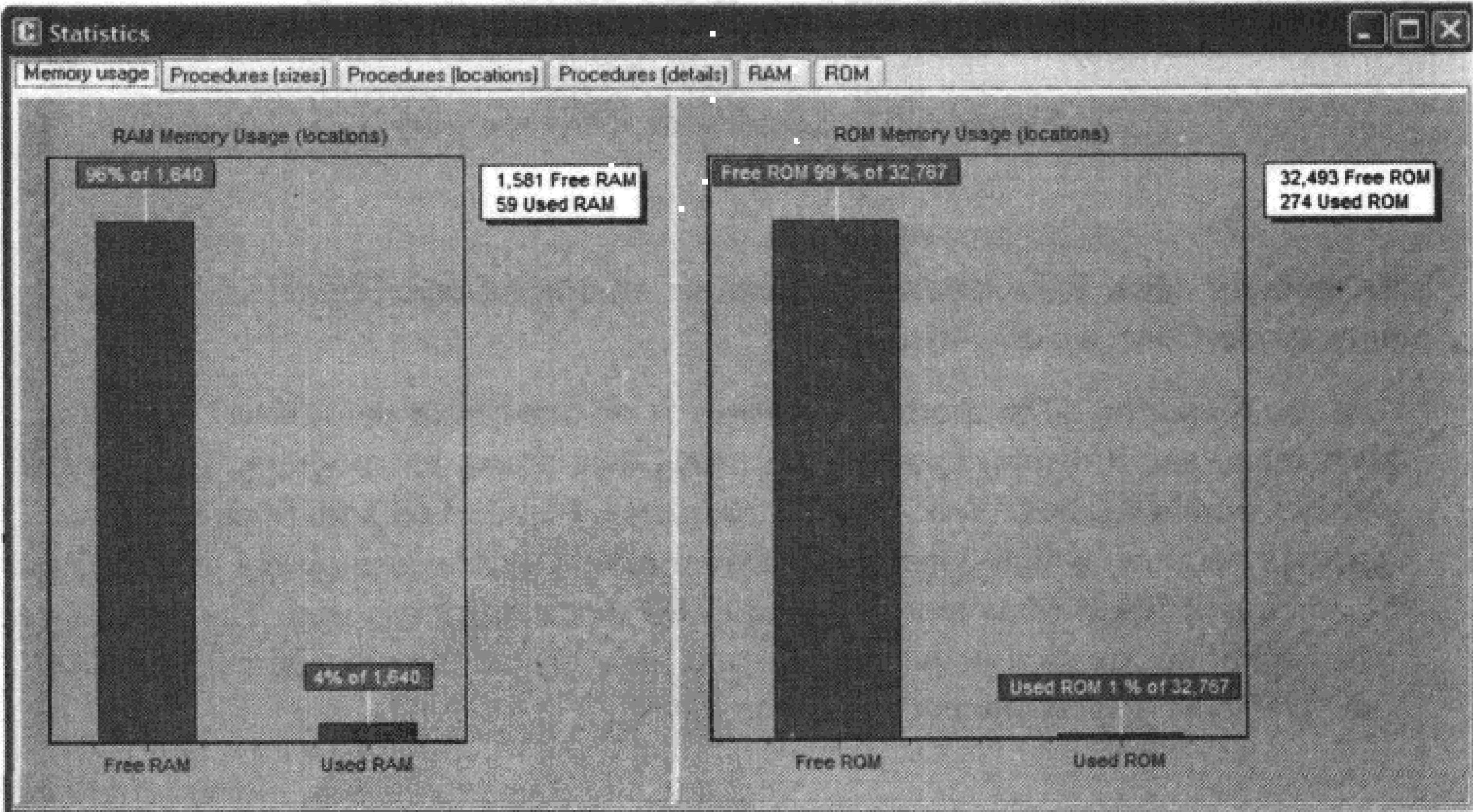


图5-63 统计窗口的显示

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

5.4 小结

本章讨论了PIC微控制器的软件开发工具（例如文本编辑器、汇编器、编译器和仿真器）和硬件开发工具（包括开发板和开发工具包、编程设备、内电路调试器和内电路模拟器）。并结合例子和项目，介绍了mikroC编译器的使用。此外，还分使用和不使用硬件的内电路调试器两种情况，介绍了基于mikroC的C程序的开发和测试步骤。最后以例子的形式讲述了在使用板上内电路调试器的情况下如何使用BIGPIC4开发板。

5.5 练习题

1. 描述微控制器系统开发周期的各个阶段。
2. 简要描述微控制器的开发工具。
3. 请说出汇编器和编译器的优缺点。
4. 请解释为什么仿真器对开发微控制器产品很有用。
5. 请详细说明什么是设备编程器。请给出PIC18系列微控制器的设备编程器的一些例子。
6. 简要描述内电路调试器和内电路模拟器的不同点。请列出两个调试工具的优缺点。
7. 在mikroC IDE中录入并编译以下的程序，改正其语法错误。然后使用软件ICD，单步运行代码，仿真程序的操作。观察仿真期间变量值的变化情况。

```
/*=====================================
              A SIMPLE LED PROJECT

This program flashes the 8 LEDs connected to PORTC of a PIC18F452
microcontroller.
=====================================*/
void main ()
{
     TRISC=0;                    // PORTC is output
     do
     {
             PORTC=0xFF;         // Turn ON LEDs on PORTC
             PORTC=0;            // Turn OFF LEDs on PORTC
             } while(1);         // Endless loop
}
```

285

8. 描述使用mikroICD内电路调试器的步骤。
9. 下面的C程序含有一些故意加入的错误。编译此程序，找出错误并改正。

```
void main()
{
     unsigned char i,j,k
     i=i0;
     j=i+1;

     for(i=0; i < 10; i++)
     {
         Sum=Sum + i;
         j++
     }
     }
}
```

10. 下面的C程序包有一些故意加入的错误。编译此程序，找出错误并改正。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
int add (int a, int b)
{
     result=a + b
}

void main()
{
       int p,q;
       p=12;
       q=I0;
       z=add(p, q)
       z++;
       for (i=0; i < z; i++)p++
}
}
```

286



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com
tyw藏书

第 6 章 简单 PIC18 项目

前面我们学习了基本接口技术和各种各样的微控制器外围寄存器，本章将会介绍简单的基于PIC18微控制器的项目设计。读者将学会使用LED、按钮开关、键盘、LED阵列、音频设备等的项目的设计，以及如何使用mikroC 编译器来开发C语言程序。尽管项目的硬件开发是在一块价格低廉的面包板上进行的，但是诸如BIGPIC4这样的开发工具包也可以用于这些项目的开发。这里，将从非常简单的项目开始，然后循序渐进地涉及复杂的项目。建议读者按照书中的顺序来学习。下面是每个项目中的内容。

- 程序描述；
- 硬件描述；
- 电路原理图；
- 算法描述（以PDL形式）；
- 程序清单；
- 进一步开发的建议。

程序的算法可以用各种各样的图表或文字方法来描述，常见的方法有流程图、结构图以及程序描述语言。本书使用程序描述语言（PDL）。

6.1 程序描述语言

程序描述语言（PDL）是一种自由格式的、类似于英文的文本，用来描述程序中的控制流。PDL不是一种编程语言，而是一个能帮助程序员在开发程序之前考虑程序逻辑的工具。现将常用的PDL关键字描述如下。

6.1.1 START-END

每个PDL程序描述（或者子程序）都以START关键字开始，以END关键字结束。PDL代码中的关键字应以加粗、大写形式出现，确保代码一目了然。同时，为了提高代码的可读性，在PDL关键字之间使用缩进是一个很好的书写习惯。

例子

```
START
    ......
    ......
END
```

6.1.2 顺序

对于程序中正常的顺序，书写语句就像书写简短的英文文本一样，感觉好像就在描述程序。

例子

```
Turn on the LED
```

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
Wait 1 second
Turn off the LED
```

6.1.3 IF-THEN-ELSE-ENDIF

使用IF、THEN、ELSE和ENDIF这些关键字来描述程序中的控制流。 288

例子

```
IF switch=1 THEN
   Turn on LED1
ELSE
   Turn on LED2
   Start the motor
ENDIF
```

6.1.4 DO-ENDDO

使用DO和ENDDO关键字来表示PDL代码中的重复操作。

例子

为了在程序中创建一个无条件循环，可以写成：

```
Turn on LED
DO 10 times
   Set clock to 1
   Wait for 10ms
   Set clock to 0
ENDDO
```

DO-ENDDO结构的变体可以使用其他的关键字（如DO-FOREVER或DO-UNTIL）来实现。请看下面的例子。

例子

为了在程序中创建一个条件循环，可以写成：

```
Turn off buzzer
IF switch=1 THEN
   DO UNTIL Port 1=1
        Turn on LED
        Wait for 10ms
        Read Port 1
   ENDDO
ENDIF
```

289

当需要一个无限循环时，可以使用下面的结构：

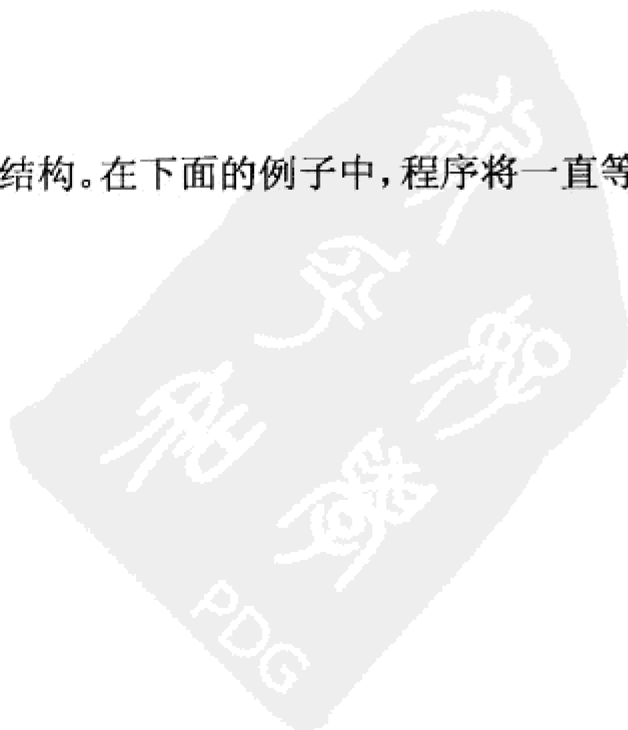
```
DO FOREVER
   Read data from Port 1
   Send data to PORT 2
   Wait for 1 second
ENDDO
```

6.1.5 REPEAT-UNTIL

REPEAT-UNTIL是在PDL代码中使用的又一个控制结构。在下面的例子中，程序将一直等待，直到开关值等于1。

例子

```
REPEAT
```



世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

```
        Turn on buzzer
        Read switch value
    UNTIL switch=1
```

注意，REPEAT-UNTIL循环至少执行一次，如果在循环结束处设置的条件不满足，那么循环的执行就会多于一次。

项目6.1　跟踪LED

项目描述

在这个项目中，8个LED被连接到PIC18F452型微控制器的PORTC端口，微控制器的工作时钟来自4 MHz的晶体振荡器。当微控制器上电（或当微控制器重启）时，LED灯按照逆时针方向轮流点亮，每次只有一只LED灯是点亮的。在相邻两次输出之间有1秒钟的延时，因此可以用肉眼看见LED灯的点亮和熄灭。

将LED灯连接至微控制器的输出端口有两种不同的模式：电流灌入模式和电流拉出模式。

电流灌入模式

如图6-1所示，在电流灌入模式中，将LED的阳极引脚连接到+5 V电源上，将阴极引脚通过一个限流电阻连接至微控制器的输出端口。

290

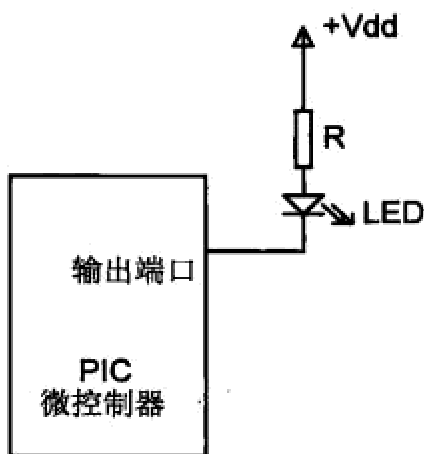


图6-1　以电流灌入模式连接LED

LED上的电压降在1.4 V到2.5 V之间的范围内波动，典型值为2 V。LED的亮度取决于流过LED的电流，该电流值在8 mA到16 mA之间波动，典型值为10 mA。

当微控制器的输出值是逻辑0时，有电流流过LED，所以LED处于点亮状态。当输出为低电平时，假设微控制器的输出电压约为0.4 V，可以使用下面的公式来计算所需的电阻值：

$$R=\frac{V_S-V_{LED}-V_L}{I_{LED}} \tag{6.1}$$

其中，

V_s是电源电压（5 V）

V_{LED}是LED两端的电压降（2 V）

V_L是当输出端为低电平时的最大输出电压（0.4 V）

I_{LED}是流过LED的电流（10 mA）

将数据代入到式（6.1），得到：

$$R=\frac{5-2-0.4}{10}=260\ \Omega$$

291

选择最接近的物理电阻值是270 Ω。

电流拉出模式

如图6-2所示，在电流拉出模式中，将LED的阳极引脚连接至微控制器的输出端口，将阴极引脚通过一个限流电阻接地。

PDG

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

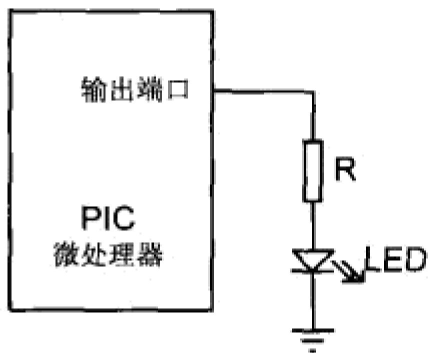


图6-2　以电流拉出模式连接LED

在这种模式中，当微控制器的输出端口是逻辑1（即+5 V）时，LED处于点亮状态。在实际应用中，输出电压大约是4.85 V，电阻值可由下式来确定：

$$R = \frac{V_O - V_{LED}}{I_{LED}} \qquad (6.2)$$

其中，V_O 是微控制器端口在逻辑值为1时的输出电压（+4.85 V）。

因此，所需的电阻值为：

$$R = \frac{4.85-2}{10} = 285\ \Omega$$

选择最接近的物理电阻值为290 Ω。

项目硬件

项目的电路原理图如图6-3所示。LED通过8个290 Ω的电阻以电流拉出模式连接到PORTC端口。将4 MHz的晶振连接到OSC1和OSC2引脚之间。另外，将一个外部的复位按钮连接至MCLR输入端口，以便在需要的时候复位微控制器。

292

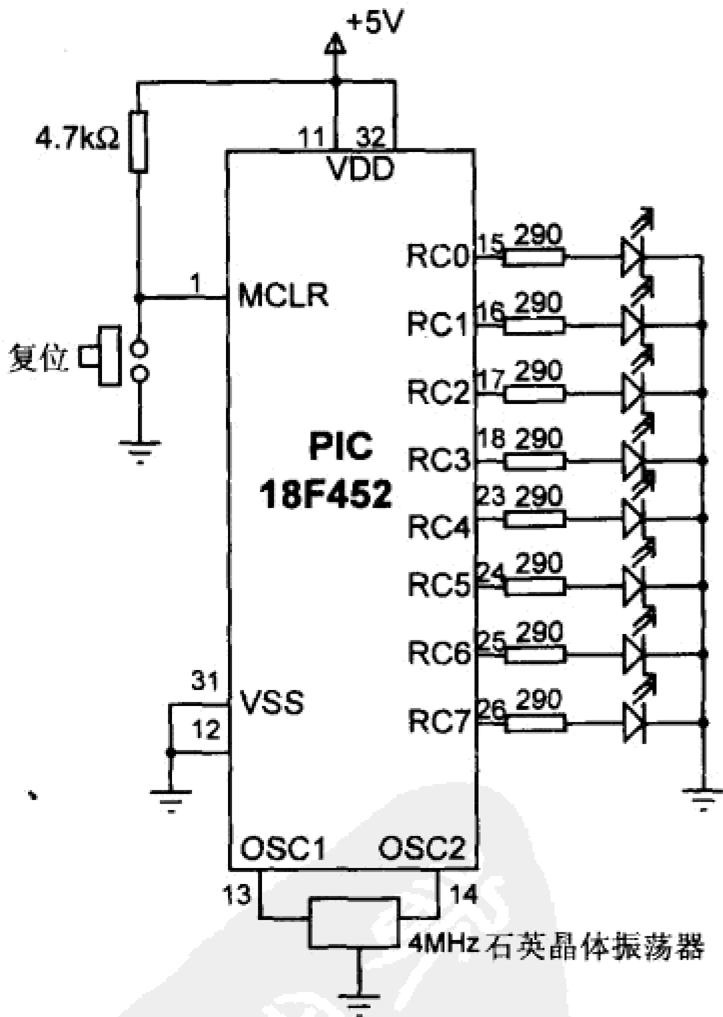
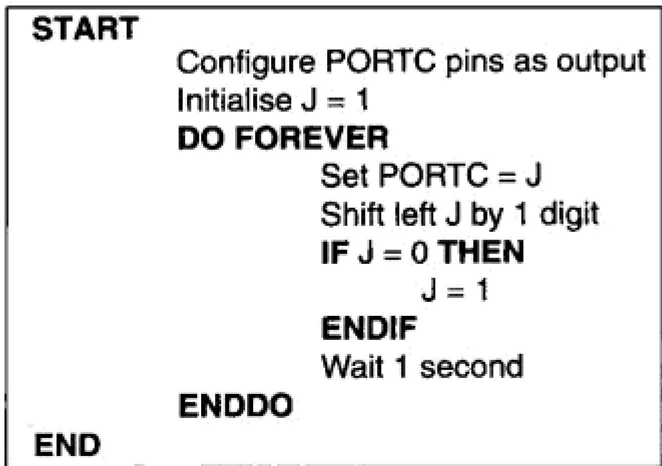


图6-3　项目的电路原理图

项目PDL

这个项目的PDL非常简单，如图6-4所示。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源



```
START
        Configure PORTC pins as output
        Initialise J = 1
        DO FOREVER
                Set PORTC = J
                Shift left J by 1 digit
                IF J = 0 THEN
                        J = 1
                ENDIF
                Wait 1 second
        ENDDO
END
```

293

图6-4　项目的PDL

项目程序

将这个程序命名为LED1.C，程序清单如图6-5所示。在程序的开始处，使用TRISC=0将PORTC引脚设定为输出端。然后形成一个无限的for循环，LED灯以逆时针方式交替点亮，从而创造出了追逐的效果。程序将连续地检测，以便当LED7点亮时，下一个将点亮的LED是LED0。

```
/****************************************************************************
                              CHASING LEDS
                              ============

In this project 8 LEDs are connected to PORTC of a PIC18F452 microcontroller
and the microcontroller is operated from a 4MHz resonator. The program turns on
the LEDs in an anti-clockwise manner with one second delay between each output.
The net result is that the LEDs seem to be chasing each other.


Author:    Dogan Ibrahim
Date:      July 2007
File:      LED1.C
*****************************************************************************/

void main()
{
   unsigned char J = 1;

   TRISC = 0;
   for(;;)                              // Endless loop
   {
       PORTC = J;                       // Send J to PORTC
       Delay_ms(1000);                  // Delay 1 second
       J = J << 1;                      // Shift left J
       if(J == 0) J = 1;                // If last LED, move to first LED
   }
}
```

图6-5　程序清单

该程序可以使用mikroC编译器来编译。项目设置应配置为：时钟信号为4 MHz，XT晶体模式，并且WDT为OFF状态。使用内电路调试器或者编程设备，将HEX文件（LED1.HEX）装载到PIC18F452微控制器中。

294

进一步开发

还可以对项目进行改进，使LED在两个方向上均能互相追逐。例如，在LED以逆时针方向移动时，可以改变方向，这样当LED的RB7处于点亮状态时，则下一个即将点亮的LED是LED RB6；当RB6处于点亮状态时，则下一个点亮的是RB5，依此类推。

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

项目 6.2 LED 骰子

项目描述

这是一个简单的骰子项目，使用一组LED、一个按钮开关和一个带有4 MHz晶振的PIC18F452微控制器。项目的方框图如图6-6所示。

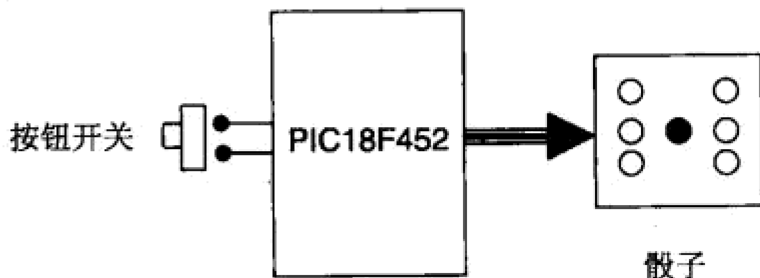


图6-6 项目的方框图

如图6-7所示，将LED排列好，当它们被点亮时，将显示一个数字，犹如一个真正的骰子一样。项目的操作如下：若所有LED都处于熄灭状态，则表示系统已经准备就绪，可以产生一个新的数字。按下按钮开关，将产生一个1~6之间的随机数，并将这个数在LED上显示3秒钟。3秒后，LED熄灭。

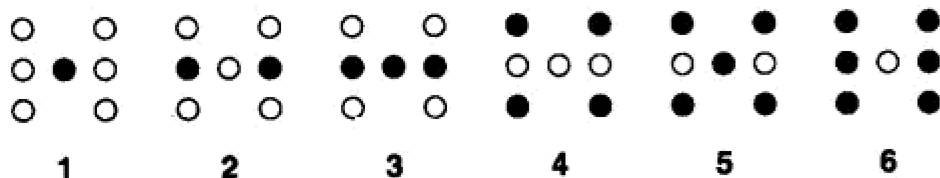


图6-7 LED骰子

295

项目硬件

项目的电路原理图如图6-8所示。7个LED表示一个骰子的面，通过290Ω的限流电阻以电流拉出模式连接至PIC18F452微控制器的PORTC端口。使用一个上拉电阻，将一个按钮开关连接到PORTB端口的位0。微控制器的工作时钟来自一个频率为4 MHz的晶振，该晶振连接在引脚OSC1和OSC2之间。使用一个+9 V的电源对微控制器供电，使用一个78L05型的电压调节器IC来获得微控制器所需要的+5 V电压。

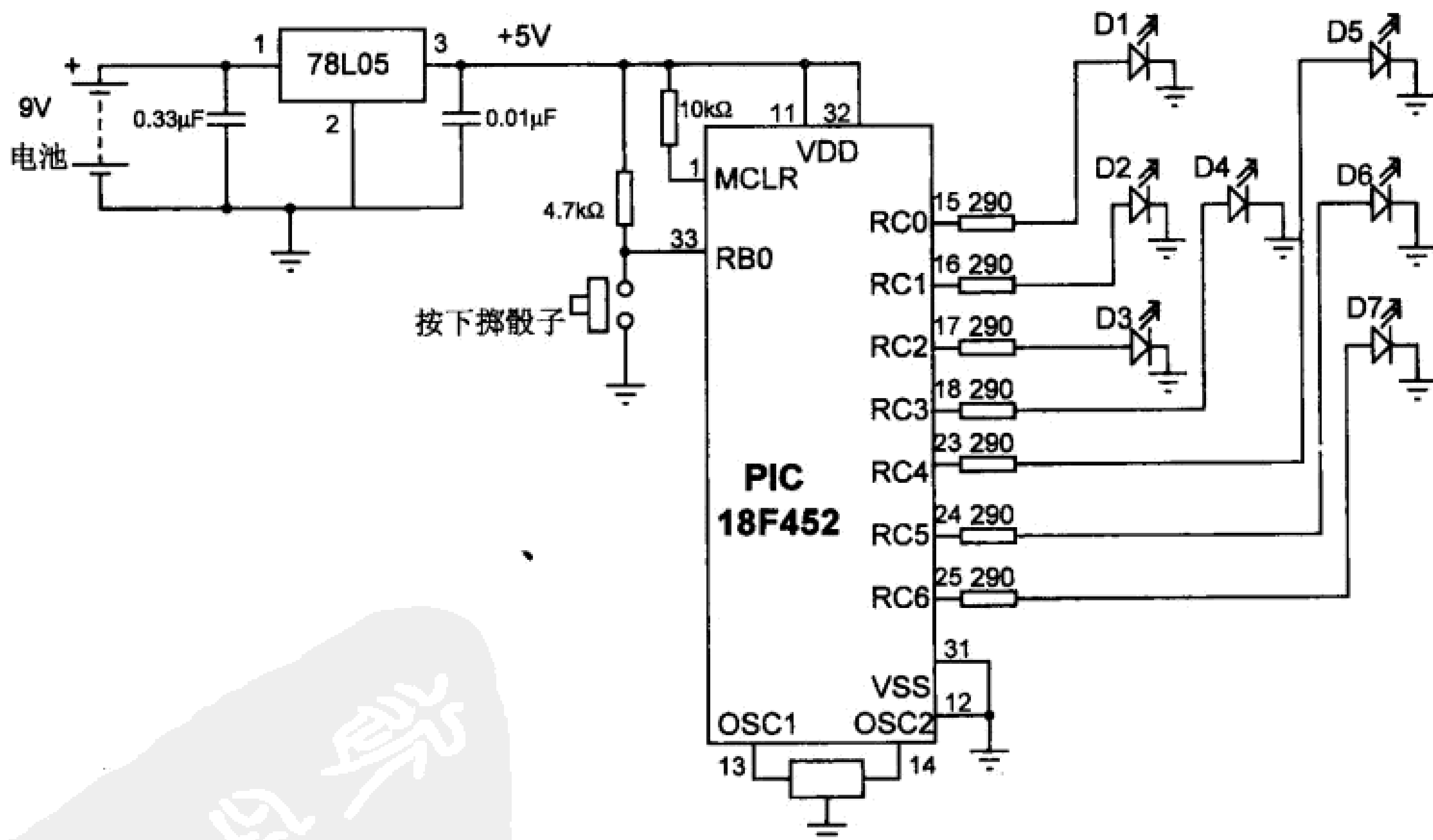


图6-8 项目的电路原理图

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

项目PDL

项目操作的PDL描述如图6-9所示。在程序的开始处，将PORTC引脚设定为输出，将PORTB的位0设定为输入。接着，程序连续地执行循环，在1到6之间依次增加变量值。按钮开关的状态将被检测，当按钮开关被按下（开关输出为逻辑0）时，当前的数字被传送至LED。使用一个简单的数组，即可找出与骰子数字相对应的应该被点亮的那些LED。

296

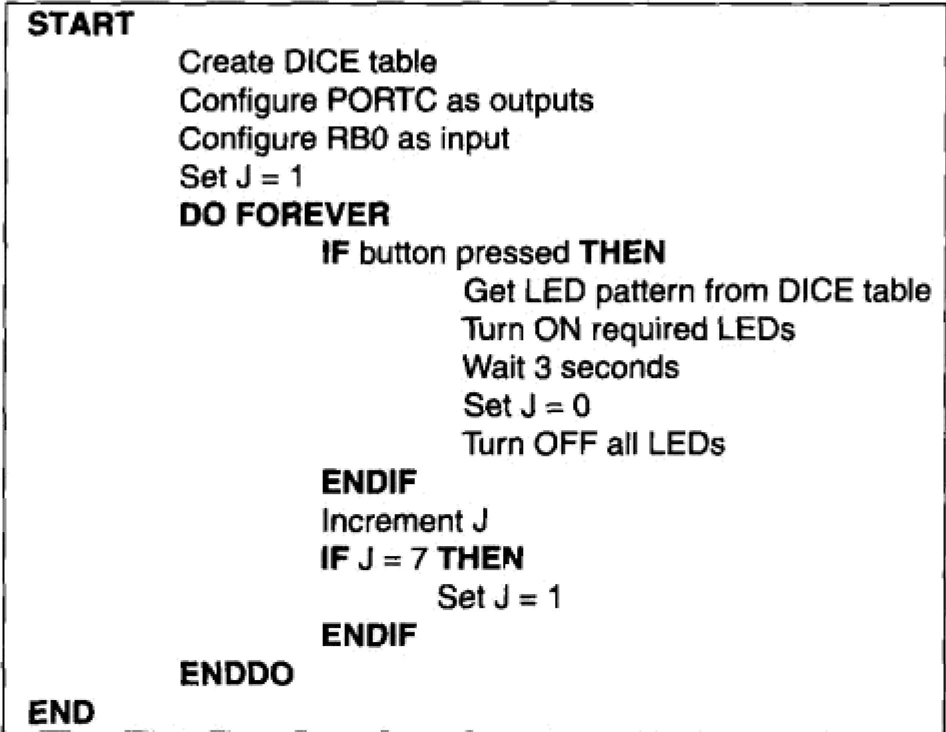


图6-9 项目PDL

表6-1给出了骰子数字与对应的处于点亮状态的、用来模拟一个真实骰子面的LED之间的关系。例如，要显示数字1（即只有中间的LED是点亮的），则必须打开D4。同样地，要显示数字4，则必须打开D1、D3、D5和D7。

表6-1 骰子数和点亮的LED

所需的数字	需要点亮的LED	所需的数字	需要点亮的LED
1	D4	4	D1，D3，D5，D7
2	D2，D6	5	D1，D3，D4，D5，D7
3	D2，D4，D6	6	D1，D2，D3，D4，D5，D6，D7

表6-2给出了所需的数字与发送到PORTC端口用以点亮正确LED的数据之间的关系。例如，要显示骰子数2，就必须发送16进制数0x22到PORTC端口。同样地，要显示数字5，就必须发送0x5D到PORTC端口，等等。

297

表6-2 所需的数字和PORTC端口数据

所需的数字	PORTC端口数据（十六进制）	所需的数字	PORTC端口数据（十六进制）
1	0x08	4	0x55
2	0x22	5	0x5D
3	0x2A	6	0x77

项目程序

将这个程序命名为LED2.C，程序清单如图6-10所示。在程序的开始处，将Switch定义为PORTB端口的位0，将Pressed 定义为0。将骰子数和要点亮的LED之间的关系存储在DICE数组中。变量J代表骰子数。变量Pattern代表发送到LED的数据。接下来，程序进入一个无限的for循环，变量J在1到6之间自动增量。当按钮开关被按下时，将从数组中读取对应于变量J当前值的LED 模式，然后发送至LED。将LED保持这

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

个状态3秒钟（使用Delay_ms函数，将参数设定为3 000ms），在此之后，将所有的LED都熄灭。然后，系统准备就绪生成一个新的骰子数。

```
/*****************************************************************************
                              SIMPLE DICE
                              ===========

In this project 7 LEDs are connected to PORTC of a PIC18F452 microcontroller
and the microcontroller is operated from a 4MHz resonator. The LEDs are organized
as the faces of a real dice. When a push-button switch connected to RB0 is pressed a
dice pattern is displayed on the LEDs. The display remains in this state for 3 seconds
and after this period the LEDs all turn OFF to indicate that the system is ready for the
button to be pressed again.

Author:    Dogan Ibrahim
Date:      July 2007
File:      LED2.C
*****************************************************************************/

#define Switch PORTB.F0
#define Pressed 0

void main()
{
   unsigned char J = 1;
   unsigned char Pattern;
   unsigned char DICE[] = {0,0x08,0x22,0x2A,0x55,0x5D,0x77};

   TRISC = 0;                          // PORTC outputs
   TRISB = 1;                          // RB0 input
   PORTC = 0;                          // Turn OFF all LEDs

   for(;;)                             // Endless loop
   {
      if(Switch == Pressed)            // Is switch pressed ?
      {
        Pattern = DICE[J];             // Get LED pattern
        PORTC = Pattern;               // Turn on LEDs
        Delay_ms(3000);                // Delay 3 second
        PORTC = 0;                     // Turn OFF all LEDs
        J = 0;                         // Initialise J
      }
      J++;                             // Increment J
      if(J == 7) J = 1;                // Back to 1 if > 6
   }
}
```

图6-10 程序清单

使用一个伪随机数生成器

在前面的项目中，变量J的值在数字1到6之间很快地变化，所以可以说产生的数字是随机的（即新的数字不依赖于过去的数字）。

伪随机数生成器函数也可以用于生成骰子数。修改过的程序清单如图6-11所示。在这个程序中，函数Number用来生成骰子数。函数接收要生成随机数的上限值（在这个例子中是6）和一个用来定义随机数集合的种子值。在本例中，种子数被设为1。每次调用函数，都将生成一个1到6之间的数。

298 ~ 300

该程序的操作基本上和图6-10所示的一样。当按钮开关被按下时，函数Number就被调用，用以生成一个新的1到6之间的骰子数，并且将这个随机数作为数组DICE的下标，用来寻找要发送至LED的位模式。

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
/*****************************************************************************
                              SIMPLE DICE
                              ===========

In this project 7 LEDs are connected to PORTC of a PIC18F452 microcontroller
and the microcontroller is operated from a 4MHz resonator. The LEDs are organized
as the faces of a real dice. When a push-button switch connected to RB0 is pressed a
dice pattern is displayed on the LEDs. The display remains in this state for 3 seconds
and after this period the LEDs all turn OFF to indicate that the system is ready for the
button to be pressed again.

In this program a pseudorandom number generator function is
used to generate the dice numbers between 1 and 6.

Author:   Dogan Ibrahim
Date:     July 2007
File:     LED3.C
*****************************************************************************/

#define Switch PORTB.F0
#define Pressed 0

//
// This function generates a pseudo random integer number
// between 1 and Lim
//
unsigned char Number(int Lim, int Y)
{
   unsigned char Result;
   static unsigned int Y;

   Y = (Y * 32719 + 3) % 32749;
   Result = ((Y % Lim) + 1);
   return Result;
}

//
// Start of MAIN program
//
void main()
{
    unsigned char J,Pattern,Seed = 1;
    unsigned char DICE[] = {0,0x08,0x22,0x2A,0x55,0x5D,0x77};

    TRISC = 0;                              // PORTC outputs
    TRISB = 1;                              // RB0 input
    PORTC = 0;                              // Turn OFF all LEDs

    for(;;)                                 // Endless loop
 {
        if(Switch == Pressed)               // Is switch pressed ?
        {
          J = Number(6,seed);               // Generate a number between 1 and 6
          Pattern = DICE[J];                // Get LED pattern
          PORTC = Pattern;                  // Turn on LEDs
          Delay_ms(3000);                   // Delay 3 second
          PORTC = 0;                        // Turn OFF all LEDs
        }
    }
}
```

图6-11 使用伪随机数生成器实现的骰子程序

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

lyw藏书

项目 6.3 双骰子项目

项目描述

本项目与项目2很相似，只是这里使用的是一对骰子（就如同很多骰子游戏一样，如西洋双陆棋），而不是一个骰子。

将图6-8中的电路原理图略加修改，为第二个骰子增加一组7个的LED。例如，第一组LED由PORTC端口驱动，第二组就由PORTD端口驱动，并且按钮开关可像之前一样连接至RB0。这样，设计需要14个输出端口用以驱动LED。稍后，将会学习到如何组合这些LED，以减少输入/输出的需求。图6-12给出了该项目的方框图。

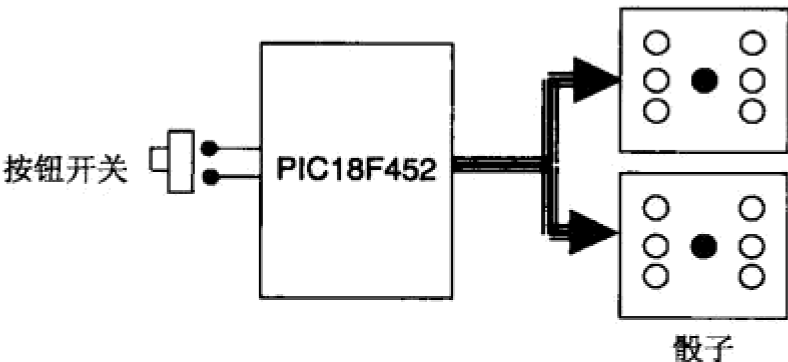


图6-12 项目的方框图

项目硬件

该项目的电路原理图如图6-13所示。电路基本上与图6-8中的电路一样，只是增加了另一组连接在PORTD端口的LED。

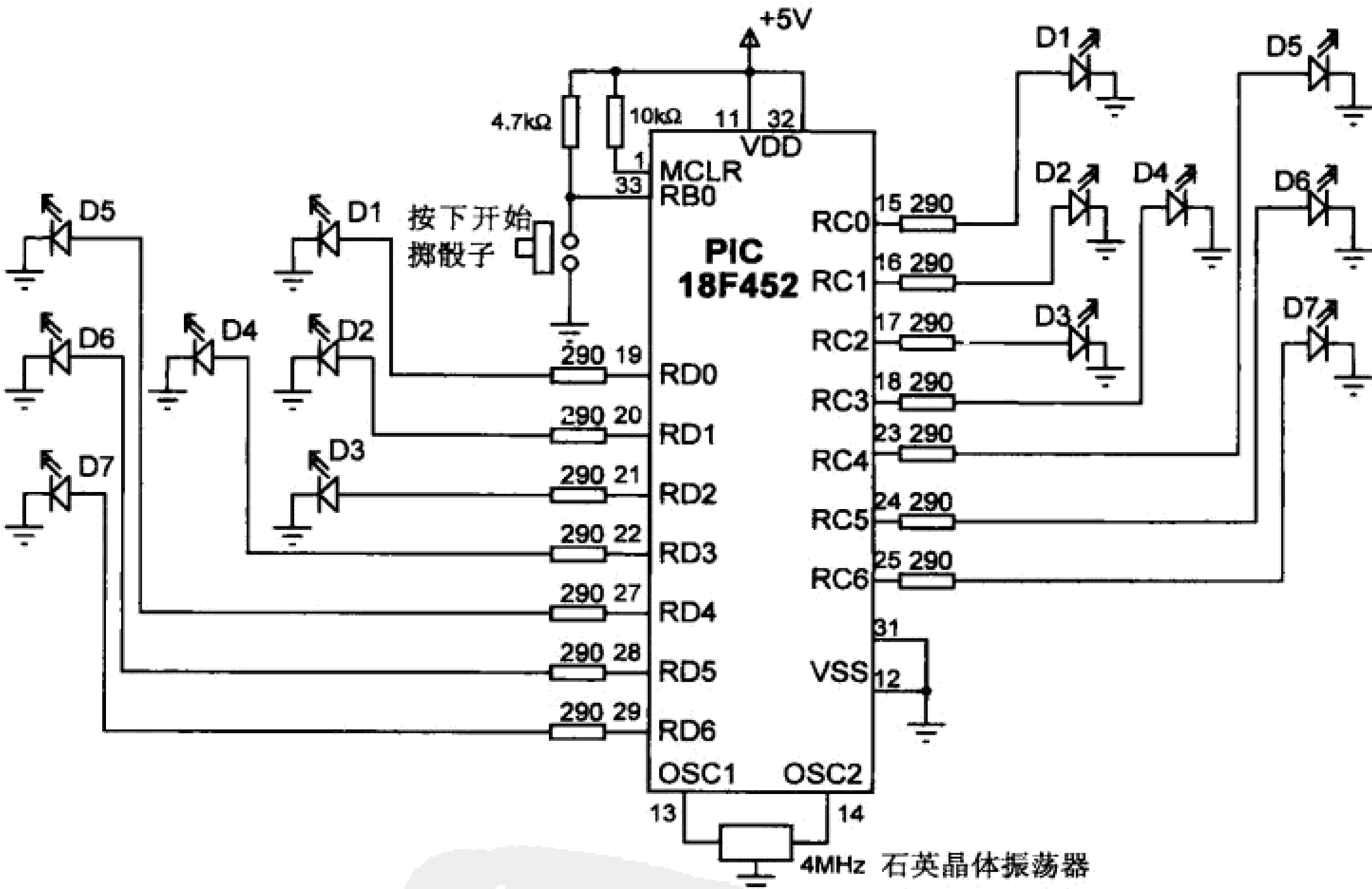


图6-13 项目的电路原理图

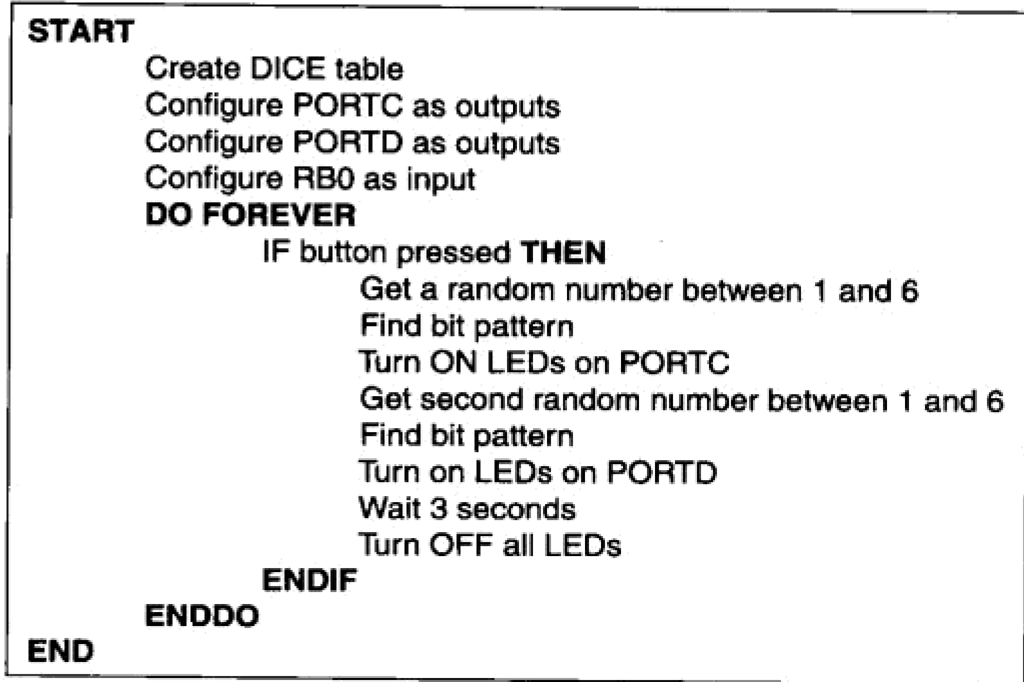
项目PDL

本项目的操作同项目2的操作非常的相似。图6-14给出了本项目的PDL描述。在程序的开始处，将

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

302 PORTC和PORTD引脚设定为输出，将PORTB（PB0）的位0设定为输到入。然后，程序在一个循环中连续地执行，并检查按钮开关的状态。当按钮开关被按下时，即生成两个1到6之间的伪随机数，并将这两个数字发送到PORTC和PORTD端口。将LED保持这个状态3秒钟，在这之后将所有的LED熄灭，用以表明若再次按下按钮开关将生成下一对数字。



```
START
        Create DICE table
        Configure PORTC as outputs
        Configure PORTD as outputs
        Configure RB0 as input
        DO FOREVER
                IF button pressed THEN
                        Get a random number between 1 and 6
                        Find bit pattern
                        Turn ON LEDs on PORTC
                        Get second random number between 1 and 6
                        Find bit pattern
                        Turn on LEDs on PORTD
                        Wait 3 seconds
                        Turn OFF all LEDs
                ENDIF
        ENDDO
END
```

图6-14 项目PDL

项目程序

将该项目程序命名为LED4.C，程序清单如图6-15所示。在程序的开始处，将Switch定义为PORTB的位0，将Pressed定义为0。同项目2中的一样，将骰子数和被点亮的LED之间的关系存储在DICE数组中。变量Pattern表示被发送至LED的数据。程序进入一个无限的for循环，在循环中不断地检查按钮开关的状态。当按钮开关被按下时，通过调用Number函数来生成两个随机数。然后确定要发送给LED的位模式，并发送到PORTC端口和PORTD端口。最后，程序在无限循环中不断地重复，检查按钮开关的状态。

```
/*************************************************************************
                              TWO DICE
                              ========

In this project 7 LEDs are connected to PORTC of a PIC18F452 microcontroller and
7 LEDs to PORTD. The microcontroller is operated from a 4MHz resonator.
The LEDs are organized as the faces of a real dice. When a push-button switch
connected to RB0 is pressed a dice pattern is displayed on the LEDs. The display
remains in this state for 3 seconds and after this period the LEDs all turn OFF to
indicate that the system is ready for the button to be pressed again.

In this program a pseudorandom number generator function is
used to generate the dice numbers between 1 and 6.

Author:   Dogan Ibrahim
Date:     July 2007
File:     LED4.C
*************************************************************************/

#define Switch PORTB.F0
#define Pressed 0

//
```

图6-15 程序清单

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
// This function generates a pseudo random integer number
// between 1 and Lim
//
unsigned char Number(int Lim, int Y)
{
   unsigned char Result;
   static unsigned int Y;

   Y = (Y * 32719 + 3) % 32749;
   Result = ((Y % Lim) + 1);
   return Result;
}


//
// Start of MAIN program
//
void main()
{
    unsigned char J,Pattern,Seed = 1;
    unsigned char DICE[] = {0,0x08,0x22,0x2A,0x55,0x5D,0x77};

    TRISC = 0;                          // PORTC are outputs
    TRISD = 0;                          // PORTD are outputs
    TRISB = 1;                          // RB0 input
    PORTC = 0;                          // Turn OFF all LEDs
    PORTD = 0;                          // Turn OFF all LEDs
    for(;;)                             // Endless loop
    {
        if(Switch == Pressed)           // Is switch pressed ?
        {
          J = Number(6,seed);           // Generate first dice number
          Pattern = DICE[J];            // Get LED pattern
          PORTC = Pattern;              // Turn on LEDs for first dice
          J = Number(6,seed);           // Generate second dice number
          Pattern = DICE[J];            // Get LED pattern
          PORTD = Pattern;              // Turn on LEDs for second dice
          Delay_ms(3000);               // Delay 3 seconds
          PORTC = 0;                    // Turn OFF all LEDs
          PORTD = 0;                    // Turn OFF all LEDS
        }
    }
}
```

图6-15 （续）

项目 6.4 使用更少的 I/O 引脚实现的两个骰子的项目

303～304

项目描述

本项目与项目3很相似，但是这里的LED是共享的，这样只需要更少的输入/输出引脚。对表6-1中的LED进行分组，结果如表6-3所示。观察这个表，可以发现：

- D4可以独立出现；
- D2和D6总是同时出现；
- D1和D3总是同时出现；
- D5和D7总是同时出现。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

表6-3 LED的分组

所需的数字	将打开的LED	所需的数字	将打开的LED
1	D4	4	D1 D3 D5 D7
2	D2 D6	5	D1 D3 D5 D7 D4
3	D2 D6 D4	6	D2 D6 D1 D3 D5 D7

305

因此，我们可以独立驱动D4，将D2、D6串联起来同时驱动，将D1、D3串联起来同时驱动，将D5、D7串联起来同时驱动（事实上，可以共享D1、D3、D5、D7，但是如果将LED串联连接，则需要8 V的驱动电压。若以并联连接的话，则需要更多的电流，并且还需要一个驱动IC）。总之，需要四条线来驱动每个骰子的7个LED。因此，两个骰子的项目可轻松地由一个8位的输出端口驱动。

项目硬件

本项目的电路原理图如图6-16所示。PIC18F452微控制器的PORTC端口可用来驱动LED，方式如下：

- 使用RC0驱动第一个骰子的D2和D6；
- 使用RC1驱动第一个骰子的D1和D3；
- 使用RC2驱动第一个骰子的D5和D7；
- 使用RC3驱动第一个骰子的D4；

306

- 使用RC4驱动第二个骰子的D2和D6；
- 使用RC5驱动第二个骰子的D1和D3；
- 使用RC6驱动第二个骰子的D5和D7；
- 使用RC7驱动第二个骰子的D4。

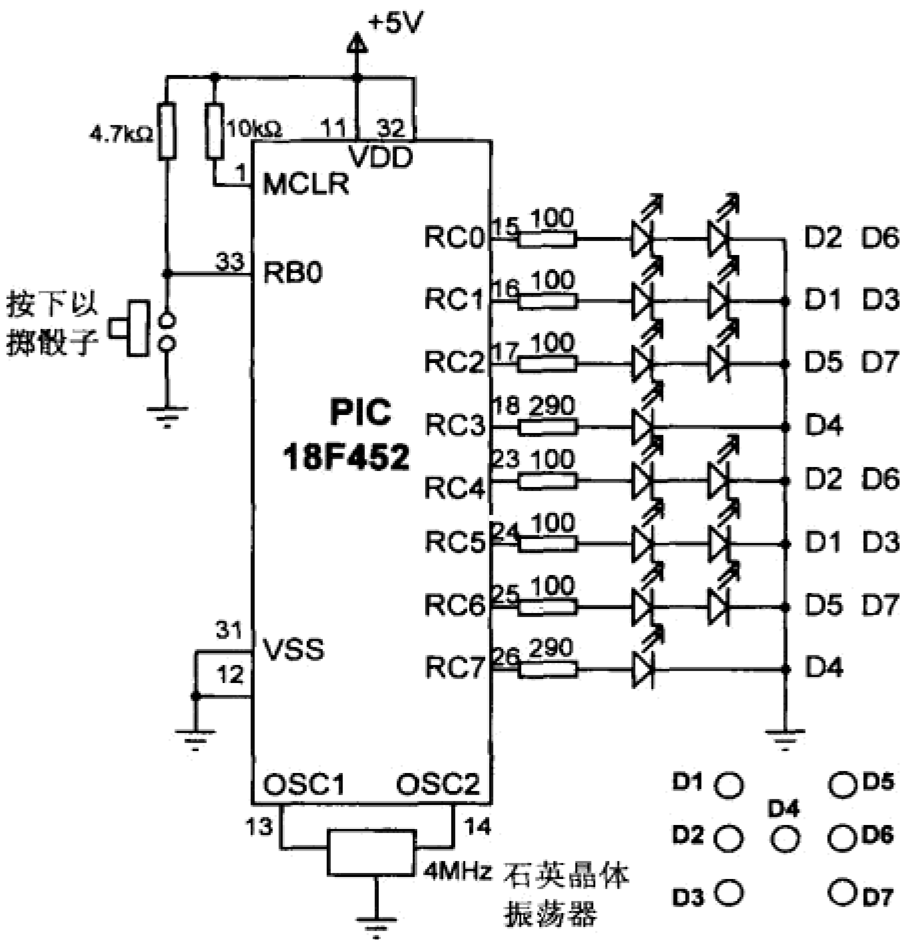


图6-16 项目的电路原理图

因为这两个LED由输出端驱动，所以可以计算出所需的限流电阻的电阻值。假设每个LED两端的电压降是2 V，流过LED的电流是10 mA，微控制器输出的高电平电压是4.85 V，则所需的电阻值为：

$$R = \frac{4.85 - 2 - 2}{10} = 85\ \Omega$$

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

于是，可以选择100Ω的电阻。

现在，需要找出骰子数和每个骰子对应的发送到LED的位模式之间的关系。表6-4给出了第一个骰子数和发送到端口引脚RC0～RC3的位模式之间的关系。同样地，表6-5给出了第二个骰子数和发送到端口引脚RC4～RC7的位模式之间的关系。

表6-4 第一个骰子的位模式

骰子数	RC3	RC2	RC1	RC0	十六进制值
1	1	0	0	0	8
2	0	0	0	1	1
3	1	0	0	1	9
4	0	1	1	0	6
5	1	1	1	0	E
6	0	1	1	1	7

表6-5 第二个骰子的位模式

骰子数	RC7	RC6	RC5	RC4	十六进制值
1	1	0	0	0	8
2	0	0	0	1	1
3	1	0	0	1	9
4	0	1	1	0	6
5	1	1	1	0	E
6	0	1	1	1	7

现在，可以找到发送到PORTC端口的8位数，用以显示两个骰子数，如下：

- 从数字生成器中得到第一个数，称之为P；
- 检索DICE表，找出低4位的位模式（即L=DICE[P]）；
- 从数字生成器中得到第二个数，称之为P；
- 检索DICE表，找出高4位的位模式（即U=DICE[P]）；
- 将高4位乘以16，然后加上低4位，找出发送到PORTC的数字（即R=16×U+L），这里R是指要发送至PORTC用来显示两个骰子数的8位数。

307

项目PDL

308

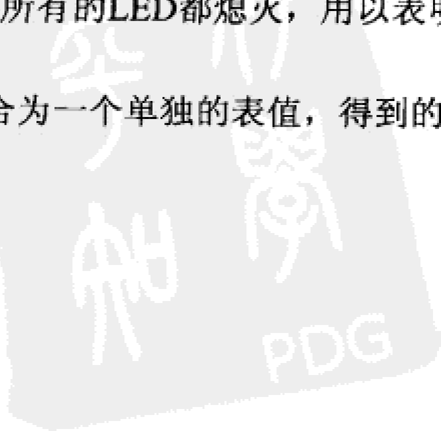
本项目的操作与项目2非常的相似。图6-17给出了该项目的PDL。在程序的开始处，将PORTC引脚设定为输出，将PORTB（RB0）的位0设定为输入。然后，程序持续地执行一个循环，并检查按钮开关的状态。当按钮开关被按下时，生成两个1到6之间的伪随机数，于是将位模式发送到PORTC端口，显示两个骰子数。保持骰子数显示3秒钟，然后将所有的LED熄灭，用以表明系统正在等待再次按下按钮开关，以显示下一组骰子数。

项目程序

将该项目的程序命名为LED5.C，程序清单如图6-18所示。在程序的开始处，将Switch定义为PORTB的位0，将Pressed定义为0。如同项目2中的一样，将骰子数和被点亮的LED之间的关系存储在一个叫作DICE的数组中。变量Pattern表示发送至LED的数据。程序进入一个无限for循环后，在循环中连续地检查按钮开关的状态。当按钮开关被按下时，通过调用Number 函数，生成两个随机数。变量L和U分别用来存储发送至PORTC端口的位模式的低四位和高四位。使用项目硬件部分描述的方法，确定要发送至PORTC的位模式，然后将其存储在变量R中。接下来，将位模式发送至PORTC端口用以同时显示两个骰子数。保持骰子数显示3秒钟，然后将所有的LED都熄灭，用以表明系统已经准备就绪。

修改程序

将两个骰子的四位整合为一个单独的表值，得到的图6-18中的程序更有效率。



世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
        START
                Create DICE table
                Configure PORTC as outputs
                Configure RB0 as input
                DO FOREVER
                        IF button pressed THEN
                                Get a random number between 1 and 6
                                Find low nibble bit pattern
                                Get second random number between 1 and 6
                                High high nibble bit pattern
                                Calculate data to be sent to PORTC
                                Wait 3 seconds
                                Turn OFF all LEDs
                        ENDIF
                ENDDO
        END
```

图6-17 项目的PDL

```
/*****************************************************************************
                TWO DICE - USING FEWER I/O PINS
                ==============================

In this project LEDs are connected to PORTC of a PIC18F452 microcontroller
and the microcontroller is operated from a 4MHz resonator. The LEDs are
organized as the faces of a real dice. When a push-button switch connected to
RB0 is pressed a dice pattern is displayed on the LEDs. The display remains
in this state for 3 seconds and after this period the LEDs all turn OFF to indicate
that the system is ready for the button to be pressed again.

In this program a pseudorandom number generator function is
used to generate the dice numbers between 1 and 6.

Author:     Dogan Ibrahim
Date:       July 2007
File:       LED5.C
*****************************************************************************/

#define Switch PORTB.F0
#define Pressed 0

//
// This function generates a pseudo random integer number
// between 1 and Lim
//
unsigned char Number(int Lim, int Y)
{
            unsigned char Result;
            static unsigned int Y;

            Y = (Y * 32719 + 3) % 32749;
            Result = ((Y % Lim) + 1);
            return Result;
}


//
// Start of MAIN program
//
void main()
```

图6-18 程序清单

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
{
    unsigned char J,L,U,R,Seed = 1;
    unsigned char DICE[] = {0,0x08,0x01,0x09,0x06,0x0E,0x07};

    TRISC = 0;                              // PORTC are outputs
    TRISB = 1;                              // RB0 input
    PORTC = 0;                              // Turn OFF all LEDs

    for(;;)                                 // Endless loop
{
        if(Switch == Pressed)               // Is switch pressed ?
        {
            J = Number(6,seed);             // Generate first dice number
            L = DICE[J];                    // Get LED pattern
            J = Number(6,seed);             // Generate second dice number
            U = DICE[J];                    // Get LED pattern
            R = 16*U + L;                   // Bit pattern to send to PORTC
            PORTC = R;                      // Turn on LEDs for both dice
            Delay_ms(3000);                 // Delay 3 seconds
            PORTC = 0;                      // Turn OFF all LEDs
        }
    }
}
```

图6-18 （续）

两个骰子的值一共有36种可能的组合方式。参阅表6-4、表6-5和图6-16，可以创建表6-6来表示所有的两个骰子值和相应的发送至PORTC端口的数。

表6-6 两骰子的组合和发送至PORTC的数据

骰 子 数	PORTC端口值	骰 子 数	PORTC端口值	骰 子 数	PORTC端口值
1, 1	0x88	3, 1	0x89	5, 1	0x8E
1, 2	0x18	3, 2	0x19	5, 2	0x1E
1, 3	0x98	3, 3	0x99	5, 3	0x9E
1, 4	0x68	3, 4	0x69	5, 4	0x6E
1, 5	0xE8	3, 5	0xE9	5, 5	0xEE
1, 6	0x78	3, 6	0x79	5, 6	0x7E
2, 1	0x81	4, 1	0x86	6, 1	0x87
2, 2	0x11	4, 2	0x16	6, 2	0x17
2, 3	0x91	4, 3	0x96	6, 3	0x97
2, 4	0x61	4, 4	0x66	6, 4	0x67
2, 5	0xE1	4, 5	0xE6	6, 5	0xE7
2, 6	0x71	4, 6	0x76	6, 6	0x77

修改过的程序（程序名为LED6.C）如图6-19所示。在这个程序中，数组DICE包含了36个可能的骰子值。程序进入一个无穷的for循环，在这个循环内将不断地检查按钮开关的状态。而且，变量从1增加到36。当按钮开关被按下时，将这个变量的值作为数组DICE的下标，用来决定发送到PORTC端口的位模式。同以前一样，保持程序显示骰子数3秒钟，然后将所有LED都熄灭，以表明系统准备就绪。

309
~
311

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
/*****************************************************************************
                    TWO DICE - USING FEWER I/O PINS
                    ==============================

In this project LEDs are connected to PORTC of a PIC18F452 microcontroller
and the microcontroller is operated from a 4MHz resonator. The LEDs are
organized as the faces of a real dice. When a push-button switch connected to
RB0 is pressed a dice pattern is displayed on the LEDs. The display remains in
this state for 3 seconds and after this period the LEDs all turn OFF to indicate
that the system is ready for the button to be pressed again.

In this program a pseudorandom number generator function is
used to generate the dice numbers between 1 and 6.

Author:   Dogan Ibrahim
Date:     July 2007
File:     LED6.C
*****************************************************************************/

#define Switch PORTB.F0
#define Pressed 0

//
// Start of MAIN program
//
void main()
{
   unsigned char Pattern, J = 1;
   unsigned char DICE[] = {0,0x88,0x18,0x98,0x68,0xE8,0x78,
                           0x81,0x11,0x91,0x61,0xE1,0x71,
                           0x89,0x19,0x99,0x69,0xE9,0x79,
                           0x86,0x16,0x96,0x66,0xE6,0x76,
                           0x8E,0x1E,0x9E,0x6E,0xEE,0x7E,
                           0x87,0x17,0x97,0x67,0xE7,0x77};

   TRISC = 0;                                 // PORTC are outputs
   TRISB = 1;                                 // RB0 input
   PORTC = 0;                                 // Turn OFF all LEDs

   for(;;)                                    // Endless loop
   {
      if(Switch == Pressed)                   // Is switch pressed ?
      {
        Pattern = DICE[J];                    // Number to send to PORTC
        PORTC = Pattern;                      // send to PORTC
        Delay_ms(3000);                       // 3 seconds delay
        PORTC = 0;                            // Clear PORTC
      }
        J++;                                  // Increment J
        if(J == 37) J = 1;                    // If J = 37, reset to 1
   }
}
```

图6-19　修改后的程序

项目 6.5　7段LED计数器

项目描述

这个项目描述了一个基于7段LED的计数器的设计，实现从0到9的连续计数，每两个计数值之间有1

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

秒的延时。本项目将阐述7段LED在PIC微控制器项目中的连接和使用。

7段LED显示器在电子电路中广泛使用，用来显示数字或字母数字值。如图6-20所示，一个7段显示器由7个基本的LED组成，这样就能显示数字0~9和一些字母。LED显示器的各段可使用字母a~g来分辨，图6-21给出了常见的7段显示器每段的名字。

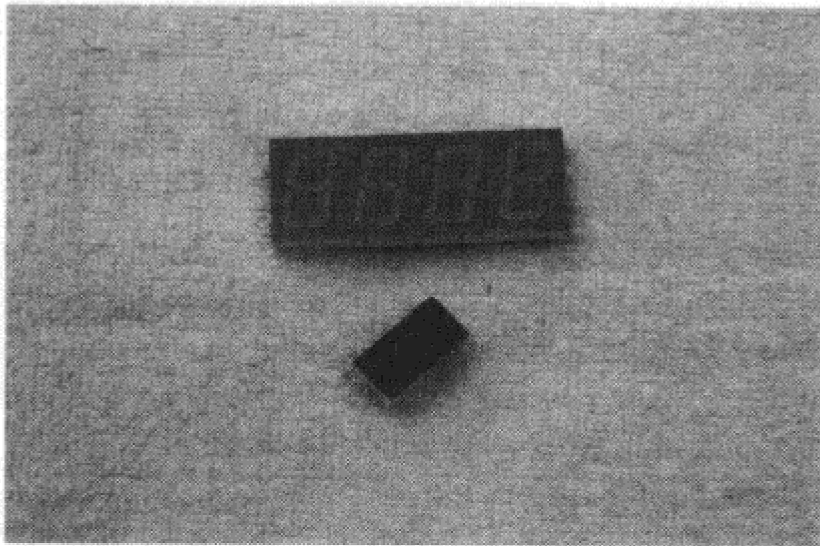


图6-20 一些7段显示器

312
~
313

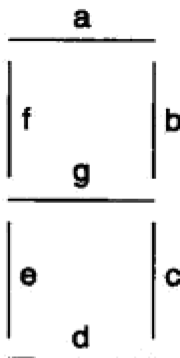


图6-21 一个7段显示器的每段段名

图6-22显示了如何打开LED显示器的不同段来获得0~9这10个数字。

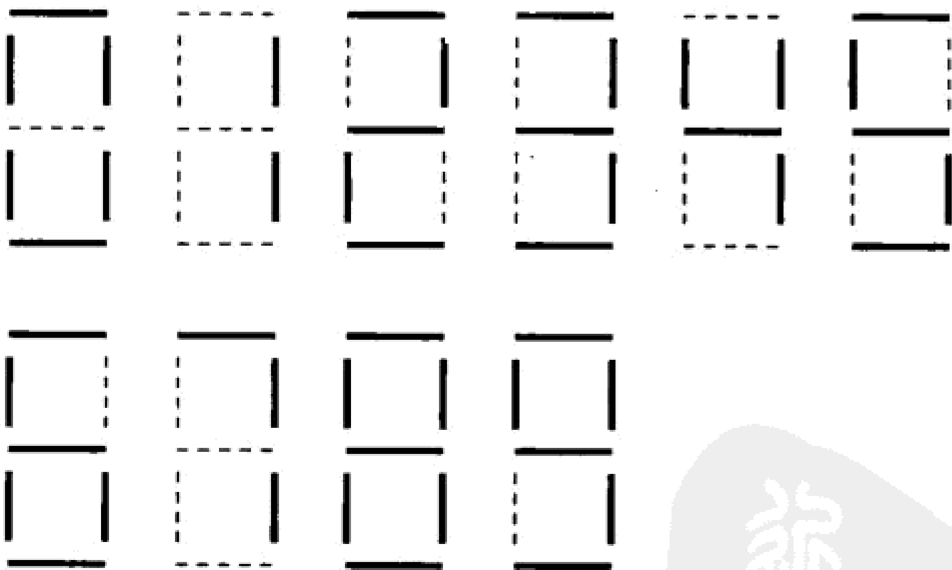


图6-22 显示数字0~9

7段显示器在两种不同配置下可用：共阴极和共阳极。如图6-23所示，在共阴极配置下，所有段LED的阴极都连接到一起并接地。通过限流电阻给目标LED段施加逻辑1，该段即被点亮。在共阴极配置中，7段LED以电流拉出模式连接到微控制器。

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

314

在共阳极配置中，所有LED的阳极端都连接在一起，如图6-24所示。然后再将公共节点连接至工作电源。将LED的阴极端通过一个限流电阻连接到逻辑0，则该端即被点亮。在共阳极配置中，7段LED以电流灌入模式连接到微控制器。

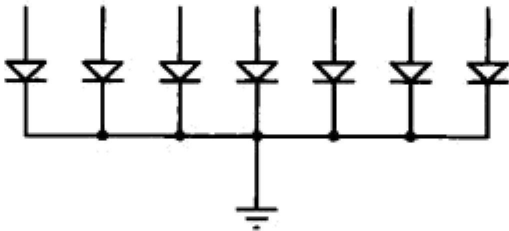


图6-23　共阴极配置

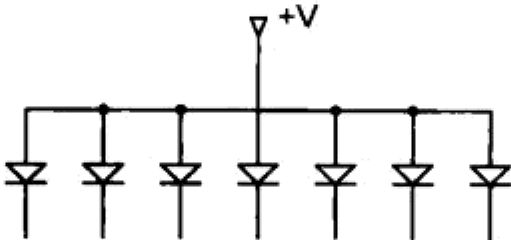


图6-24　共阳极配置

在这个项目中，使用的是一个Kingbright SA52-11 红色共阳极7段显示器。这是一个13 mm（0.52英寸）的显示器，共有10个引脚，其中包括一个用来显示小数点的LED段。表6-7给出了这种显示器的引脚配置。

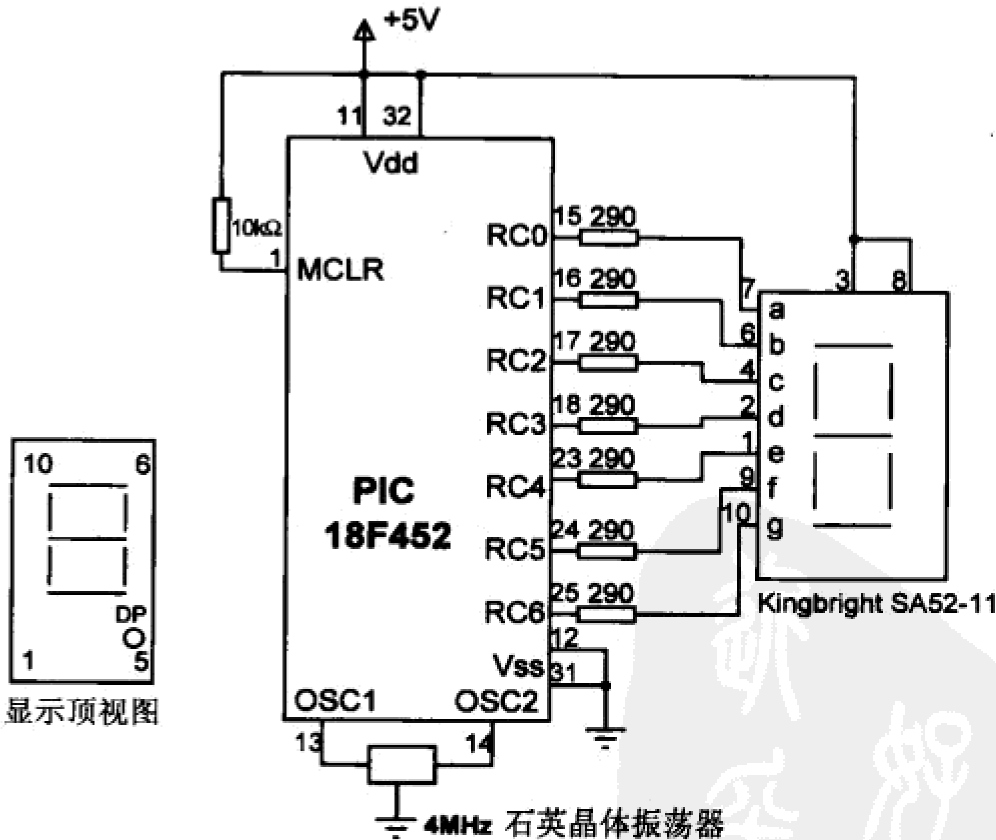
表6-7　SA52-11引脚配置

引脚编号	段 描 述	引脚编号	段 描 述
1	e	6	b
2	d	7	a
3	共阳极	8	共阳极
4	c	9	f
5	小数点	10	g

项目硬件

该项目的电路原理图如图6-25所示。PIC18F452型微控制器使用一个4 MHz的晶体振荡器。显示器的a段到g段通过290Ω的限流电阻连接到微控制器的PORTC端口。在驱动显示器之前，必须知道要显示的数字与对应的被点亮的LED段之间的关系，如表6-8所示。例如，要显示数字3，必须发送十六进制数0x4F到PORTC端口，这样就会将a、b、c、d和g段点亮。同样地，要显示数字9，必须发送十六进制数0x6F到PORTC端口，这样就会将a、b、c、d、f和g段点亮。

315



316

图6-25　项目的电路原理图

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

表6-8 显示的数字和发送到PORTC的数据

数 字	x g f e d c b a	PORTC数据
0	0 0 1 1 1 1 1 1	0x3F
1	0 0 0 0 0 1 1 0	0x06
2	0 1 0 1 1 0 1 1	0x5B
3	0 1 0 0 1 1 1 1	0x4F
4	0 1 1 0 0 1 1 0	0x66
5	0 1 1 0 1 1 0 1	0x6D
6	0 1 1 1 1 1 0 1	0x7D
7	0 0 0 0 0 1 1 1	0x07
8	0 1 1 1 1 1 1 1	0x7F
9	0 1 1 0 1 1 1 1	0x6F

没有使用x，用0代替

项目PDL

该项目操作的PDL描述如图6-26所示。在程序的开始处，声明一个叫作SEGMENT的数组，并将数字0~9和发送至PORTC端口的数据之间的关系存储在该数组中。然后，将PORTC引脚设置为输出，将变量初始化为0。接下来，程序进入一个无穷的循环，在这个循环中变量在0~9中进行增量，相应的用来点亮正确LED段的位模式被不断地发送给PORTC端口，在每两次输出之间有1秒的延时。 [317]

```
START
        Create SEGMENT table
        Configure PORTC as outputs
        Initialize CNT to 0
        DO FOREVER
                Get bit pattern from SEGMENT corresponding to CNT
                Send this bit pattern to PORTC
                Increment CNT between 0 and 9
                Wait 1 second
        ENDDO
END
```

图6-26 项目的PDL

项目程序

将该项目程序命名为SEVEN1.C，其程序清单如图6-27所示。在程序的开始处，将变量`Pattern`和变量`Cnt`都声明为字符类型，并将变量`Cnt`清零。表6-8是使用数组SEGMENT实现的。当将PORTC引脚设置为输出后，程序进入一个无穷的`for`循环。在这个循环里，将会发现与`Cnt`内容相对应的位模式，然后将其存储在变量`Pattern`中。因为使用的是共阳极显示器，当输入为逻辑0时，要求该段LED即被点亮，因此应在发送至PORTC之前将位模式取反。变量`Cnt`的值在0~9之间自动增量，在下次重复上述顺序之前需要等待1秒钟。 [318]

```
/*****************************************************************************
                        7-SEGMENT DISPLAY
                        =================

In this project a common anode 7-segment LED display is connected to PORTC
of a PIC18F452 microcontroller and the microcontroller is operated from a 4MHz
resonator. The program displays numbers 0 to 9 on the display with a one second
delay between each output.

Author:    Dogan Ibrahim
```

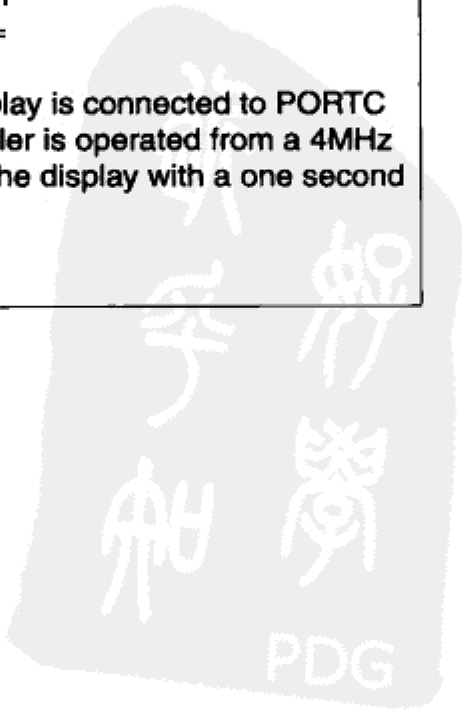


图6-27 程序清单

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com


```
Date:     July 2007
File:     SEVEN1.C
*****************************************************************************/

void main()
{
   unsigned char Pattern, Cnt = 0;
   unsigned char SEGMENT[] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,
                              0x7D,0x07,0x7F,0x6F};

   TRISC = 0;                        // PORTC are outputs

   for(;;)                           // Endless loop
   {
     Pattern = SEGMENT[Cnt];         // Number to send to PORTC
     Pattern = ~Pattern;             // Invert bit pattern
     PORTC = Pattern;                // Send to PORTC
     Cnt++;
     if(Cnt == 10) Cnt = 0;          // Cnt is between 0 and 9
     Delay_ms(1000);                 // 1 second delay
   }
}
```

图6-27　（续）

修改程序

注意到，如果创建一个用来显示数字的函数，然后在主程序中调用这个函数，那么程序的可读性会变得更强。修改过的程序如图6-28（被命名为SEVEN2.C）所示。这里，创建了一个叫作Display的函数，其中参数为no。该函数从使用no作为下标的本地数组SEGMENT中得到位模式，然后将位模式的结果返回给调用程序。

```
/*****************************************************************************
                     7-SEGMENT DISPLAY
                     =================

In this project a common anode 7-segment LED display is connected to
PORTC of a PIC18F452 microcontroller and the microcontroller is
operated from a 4MHz resonator. The program displays numbers 0 to 9 on
the display with a one second delay between each output.

In this version of the program a function called "Display" is used to display the
number.

Author:   Dogan Ibrahim
Date:     July 2007
File:     SEVEN2.C
*****************************************************************************/

//
// This function displays a number on the 7-segment LED.
// The number is passed in the argument list of the function.
//
unsigned char Display(unsigned char no)
{
   unsigned char Pattern;
   unsigned char SEGMENT[] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,
                    0x7D,0x07,0x7F,0x6F};
```

图6-28　修改后的程序清单

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com


```
    Pattern = SEGMENT[no];
    Pattern = ~ Pattern;                         // Pattern to return
    return (Pattern);
}


//
// Start of MAIN Program
//
void main()
{
    unsigned char Cnt = 0;

    TRISC = 0;                                   // PORTC are outputs

    for(;;)                                      // Endless loop
    {
      PORTC = Display(Cnt);                      // Send to PORTC
      Cnt++;
      if(Cnt == 10) Cnt = 0;                     // Cnt is between 0 and 9
      Delay_ms(1000);                            // 1 second delay
    }
}
```

图6-28 （续）

项目 6.6 两个数位的多路复用 7 段 LED

项目描述

该项目与项目6.5很相似，但是这里使用的是多路复用的两个数位，而不是只有一位数位且是固定的数字。在这个项目中，显示的数为25。在多路复用LED应用（如图6-29所示）中，所有数位的LED段都连接在一起，并且每个数位的共用引脚都由微控制器分别打开。当每个数位都仅仅显示几毫秒时，使用肉眼根本看不出来数位不是一直处于点亮状态。采用这样方式，就能多路复用7段显示器的任何数字。例如，要显示数53，则必须将5发送到第一个数位，且使其共用引脚可用。在几毫秒之后，将数字3发送至第二个数位，且使其共用引脚变为可用。连续地重复这个过程，它带给用户的感觉就是两个显示器都一直处于点亮状态。

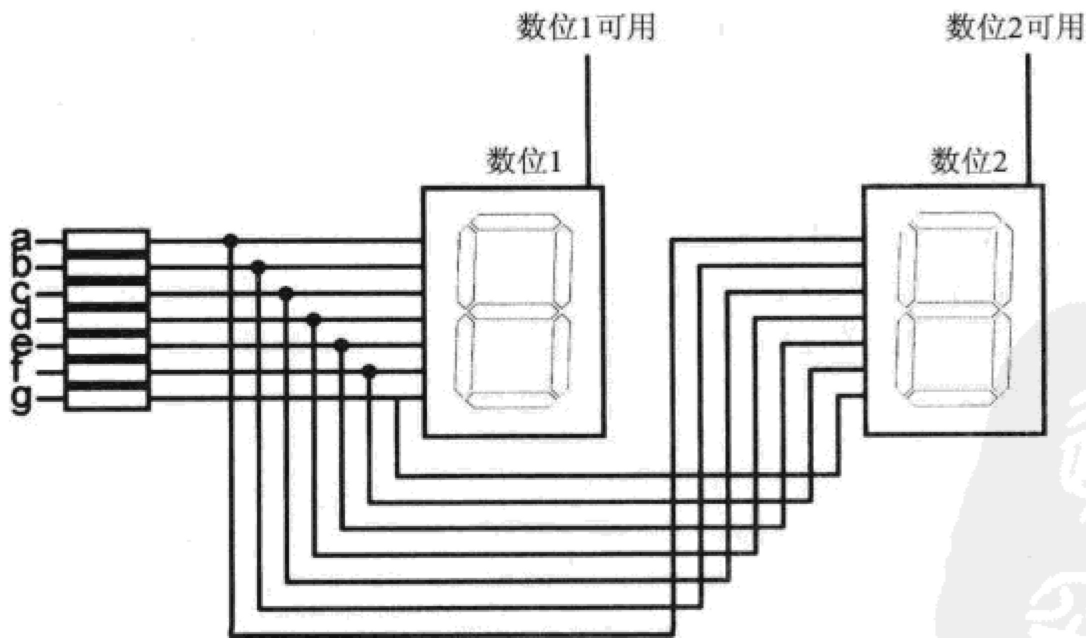


图6-29 两个多路复用的7段显示器

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

319~320

有些制造商还提供独立封装形式的多路复用的多数位显示器，如2位、4位或者8位的多路复用显示器。在这个项目中，使用的显示器是DC56-11EWA，这是一个红色0.56英寸共阳极的两数位显示器，有18个引脚，其引脚配置如表6-9所示。微控制器可按如下的方式控制该显示器：

- 对于数位1，将段的位模式发送到a至g段；
- 使能数位1；
- 等待几毫秒；
- 禁止数位1；
- 对于数位2，将段的位模式发送到a至g段；
- 使能数位2；
- 等待几毫秒；
- 禁止数位2；
- 连续地重复以上步骤。

表6-9 DC56-11EWA双数位显示器的引脚设置

引脚编号	段名字	引脚编号	段名字
1,5	E	16,11	A
2,6	D	18,12	F
3,8	C	13	数位2可用
14	数位1可用	4	小数点1
17,7	G	9	小数点2
15,10	B		

DC56-11EWA显示器的段设置如图6-30所示。在多路复用显示器应用中，相应段的段引脚是连接在一起的。例如，将引脚11和引脚16连接在一起，作为公共的a段，将引脚15和引脚10连接在一起，作为公共的b段，依此类推。

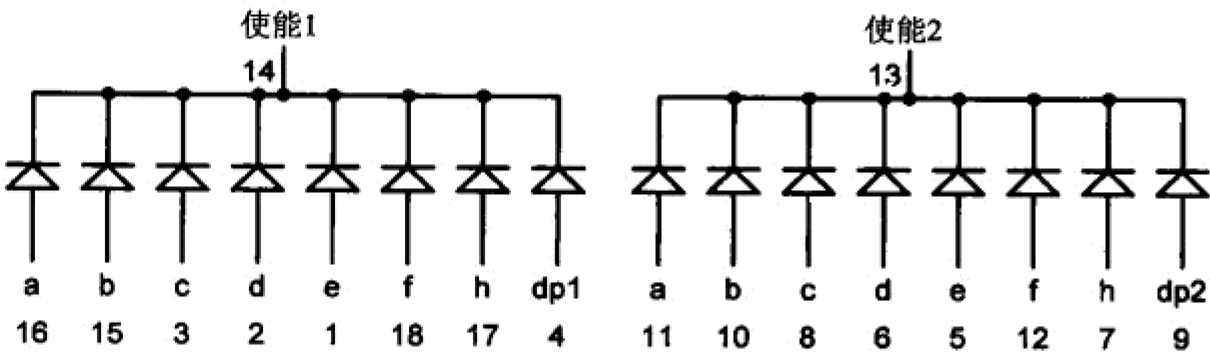


图6-30 DC56-11EWA显示器的段设置

项目硬件

该项目的方框图如图6-31所示，其电路原理图如图6-32所示。将显示器的段连接到PIC18F452型微控制器的PORCT端口，使用4 MHz的晶体振荡器。显示器的每一段都连接有限流电阻。为了使能每个数位，这里使用一个BC108型的晶体管开关连接至微控制器的端口引脚RB0和RB1。当向晶体管的基极施加逻辑1时，相应的段就被点亮了。

321~322

项目PDL

在程序的开始处，将PORTB和PORTC引脚设置为输出。接着，程序进入一个无穷的循环，在循环中，首先计算数的最高有效位（MSD），然后调用函数Display来确定位模式，再将位模式发送至显示器，并将数位1置为可用。经过短暂的延时后，将数位1置为不可用，计算数的最低有效位（LSD），然后调用函数Display来确定位模式，然后将位模式发送至显示器，并将数位2置为可用。又经过很短的延时，将数位2置为不可用。将这个过程无限地重复。图6-33给出了该项目的PDL。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

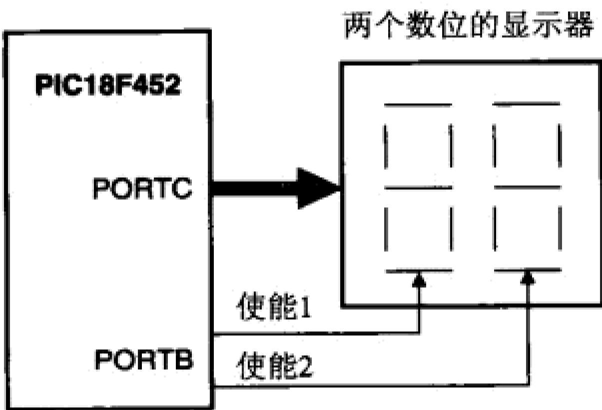


图6-31 项目的方框图

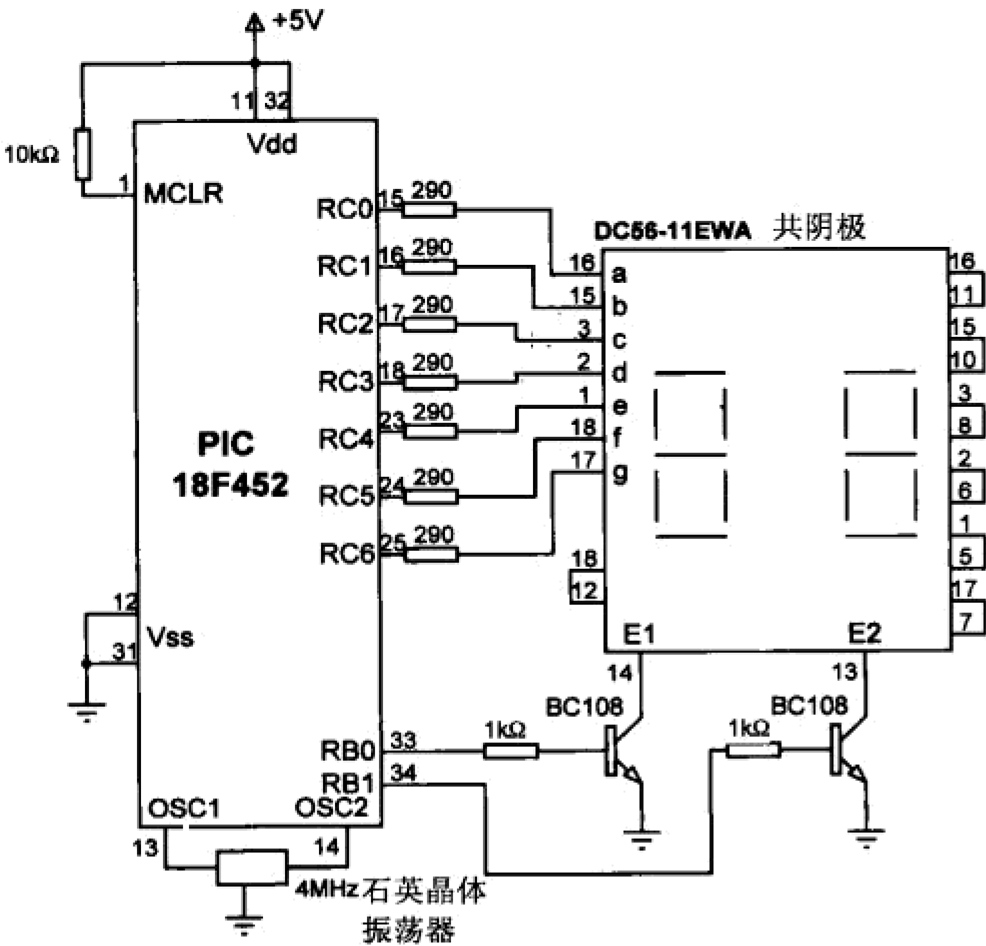
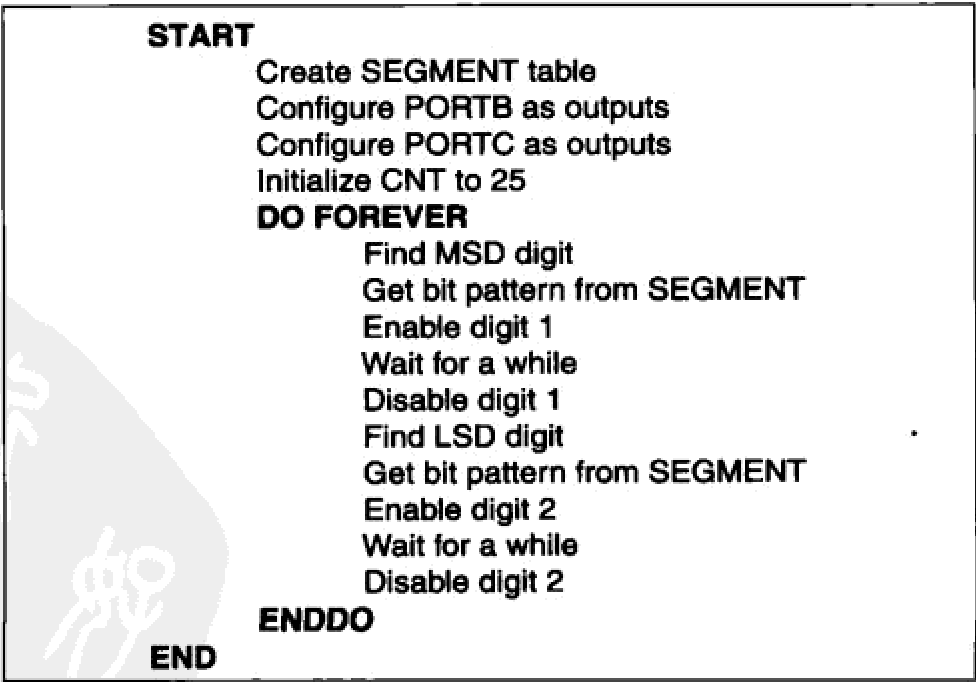


图6-32 项目的电路原理图



```
START
        Create SEGMENT table
        Configure PORTB as outputs
        Configure PORTC as outputs
        Initialize CNT to 25
        DO FOREVER
                Find MSD digit
                Get bit pattern from SEGMENT
                Enable digit 1
                Wait for a while
                Disable digit 1
                Find LSD digit
                Get bit pattern from SEGMENT
                Enable digit 2
                Wait for a while
                Disable digit 2
        ENDDO
END
```

图6-33 项目PDL

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

lyw藏书

项目程序

将该程序命名为SEVEN3.C，程序清单如图6-34所示。将DIGIT1和DIGIT2 分别定义为PORTB端口的位0和位1。将需要显示的值（数25）存储在变量Cnt中。使用for语句形成一个无穷的循环。在循环中，将数的最高有效位（MSD）除以10，然后调用函数Display来确定要发送至PORTC的位模式。然后使用DIGIT1=1，将数位1置为可用，此时程序等待10 ms。在这之后，将数位1置为不可用，然后使用模运算符（%）来确定数的最低有效位（LSD），并将结果发送至PORTC端口。同时，使用DIGIT2=1，将数位2置为可用，并让程序等待10 ms。在这之后，将数位2置为不可用，并且让程序永远地重复这个过程。

323~325

```
/************************************************************************
                    Dual 7-SEGMENT DISPLAY
                    =====================

In this project two common cathode 7-segment LED displays are connected to
PORTC of a PIC18F452 microcontroller and the microcontroller is operated
from a 4MHz resonator. Digit 1 (left digit) enable pin is connected to port pin
RB0 and digit 2 (right digit) enable pin is connected to port pin RB1 of the
microcontroller. The program displays number 25 on the displays.

Author:   Dogan Ibrahim
Date:     July 2007
File:     SEVEN3.C
*************************************************************************/
#define DIGIT1 PORTB.F0
#define DIGIT2 PORTB.F1

//
// This function finds the bit pattern to be sent to the port to display a number
// on the 7-segment LED. The number is passed in the argument list of the function.
//
unsigned char Display(unsigned char no)
{
   unsigned char Pattern;
   unsigned char SEGMENT[] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,
                              0x7D,0x07,0x7F,0x6F};

   Pattern = SEGMENT[no];                     // Pattern to return
   return (Pattern);
}


//
// Start of MAIN Program
//
void main()
{
   unsigned char Msd, Lsd, Cnt = 25;

   TRISC = 0;                                 // PORTC are outputs
   TRISB = 0;                                 // RB0, RB1 are outputs

   DIGIT1 = 0;                                // Disable digit 1
   DIGIT2 = 0;                                // Disable digit 2

   for(;;)                                    // Endless loop
   {
```

图6-34 程序清单

PDG

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com


```
Msd = Cnt / 10;                         // MSD digit
    PORTC = Display(Msd);               // Send to PORTC
    DIGIT1 = 1;                         // Enable digit 1
    Delay_Ms(10);                       // Wait a while

    DIGIT1 = 0;                         // Disable digit 1
    Lsd = Cnt % 10;                     // LSD digit
    PORTC = Display(Lsd);               // Send to PORTC
    DIGIT2 = 1;                         // Enable digit 2
    Delay_Ms(10);                       // Wait a while
    DIGIT2 = 0;                         // Disable digit 2
  }
}
```

图6-34 （续）

项目 6.7 带定时器中断的两数位多路复用 7 段 LED 计数器

项目描述

这个项目与项目6很相似，但是这里微控制器的定时器中断是用来刷新显示器的。在项目6中，微控制器忙于每10 ms更新一次显示器，并不能执行任何其他的任务。例如，项目6中给出的程序不能用来生成计数器，实现在每两次计数之间延时1秒，因为程序在等待的1秒钟时间里是不能更新显示器的。

在项目中，设计了一个计数器，用来从0计数到99，使用定时器中断服务子程序，每5 ms刷新一次显示器。然后，主程序可以执行其他的任务，在这个例子中是增量计数，并在两次计数之间等待1 s。

在项目中，定时器Timer0工作在8位模式。中断时间的计算可按下式进行：

$$时间=（4\times 时钟周期）\times Prescaler\times（256-TMR0L）$$

其中，Prescaler是选择的预分频器值，TMR0L是加载给定时器寄存器TMR0L的值，用来在每个Time周期生成定时器中断。

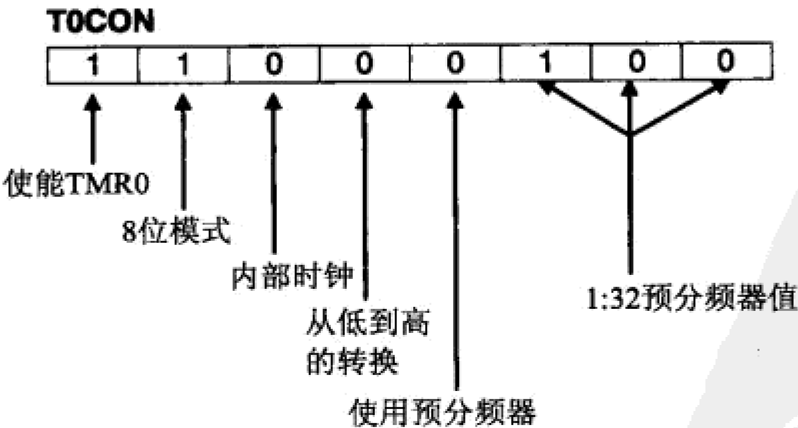
在我们的应用中，时钟频率是4 MHz，也就是，时钟周期=0.25 μs，并且时间=5 ms。选择预分频器值为32，可以计算出需要加载到TMR0L的数是： 326

$$TMR0L = 256 - \frac{时间}{4\times 时钟周期\times prescaler}$$

或者

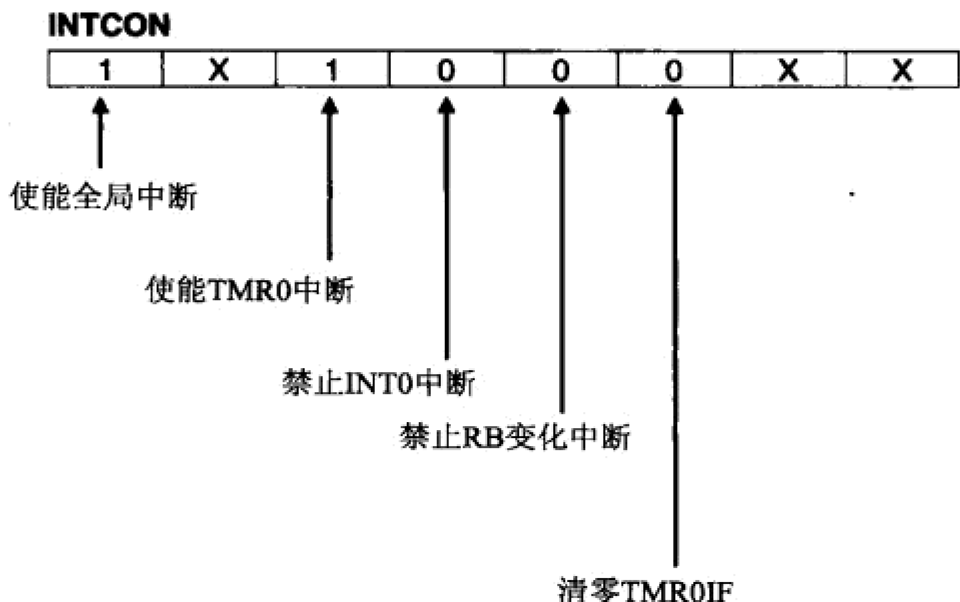
$$TMR0L = 256 - \frac{5000}{40\times 0.25\times 32} = 100$$

因此，TMR0L被装载的值为100。加载到TMR0控制寄存器T0CON的值如下：



因此，应向T0CON寄存器装载的十六进制数是0xC4。下一个要设置的寄存器是中断控制寄存器INTCON，在这里我们禁止使用基于优先级的中断，使能全局中断和TMR0中断： 327

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com



将不关心的位(X)当作0，装载到寄存器INTCON的十六进制数为0xA0。

当中断发生时，程序将自动地跳转到中断服务子程序。在子程序内部，必须重新装载寄存器TMR0L，重新使能TMR0中断，并且将TMR0的中断标志位清零。设置INTCON寄存器为0x20，重新使能TMR0中断，同时将TMR0的中断标志清零。

因而，需要执行的操作可以总结如下：

- 装载TMR0L为100；
- 设置T0CON为0xC4；
- 设置INTCON为0xA0；
- 增量计数，每次延时1秒。

在中断服务子程序中：

- 重新装载TMR0L为100；
- 刷新显示器；
- 设置INTCON为0x20（重新使能TMR0中断，并清零定时器的中断标志）。

项目硬件

该项目的电路原理图与图6-32中的一样，将一个双7段的显示器连接到PIC18F452微控制器的PORTB和PORTC端口。

项目PDL

该项目的PDL描述如图6-35所示。程序分为两个部分：主程序和中断服务子程序。在主程序中，TMR0被设置为每5ms生成一次中断，在延时1秒后计数器的值自动增量。在中断服务子程序中，重新使能定时器的中断，并且每隔5 ms刷新一次显示器。

328

项目程序

将该程序命名为SEVEN4.C，程序清单如图6-36所示。在主程序的开始处，将PORTB和PORTC端口设定为输出。然后向寄存器T0CON装载0xC4，使能TMR0，将预分频器值设为32。向TMR0L寄存器装载100，以每隔5ms生成一个中断。然后程序进入一个无穷的循环，在循环中变量cnt的值每隔1秒增

```
MAIN PROGRAM:

     START
          Configure PORTB as outputs
          Configure PORTC as outputs
          Clear variable Cnt to 0
          Configure TMR0 to generate interrupts every 5ms
          DO FOREVER
               Increment Cnt between 0 and 99
               Delay 1 second
          ENDO
     END


INTERRUPT SERVICE ROUTINE:
     START
          Re-configure TMR0
          IF Digit 1 updated THEN
                 Update digit 2
          ELSE
                 Update digit 1
          END
     END
```

图6-35　项目的PDL

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

量一次。

```
/*************************************************************************
                    Dual 7-SEGMENT DISPLAY COUNTER
                    ==============================

In this project two common cathode 7-segment LED displays are connected to
PORTC of a PIC18F452 microcontroller and the microcontroller is operated
from a 4MHz resonator. Digit 1 (left digit) enable pin is connected to port pin RB0
and digit 2 (right digit) enable pin is connected to port pin RB1 of the microcontroller.
The program counts up from 0 to 99 with one second delay between each count.

The display is updated in a timer interrupt service routine at
every 5ms.

Author:   Dogan Ibrahim
Date:     July 2007
File:     SEVEN4.C
*************************************************************************/
#define DIGIT1 PORTB.F0
#define DIGIT2 PORTB.F1

unsigned char Cnt = 0;
unsigned char Flag = 0;

//
// This function finds the bit pattern to be sent to the port to display a number
// on the 7-segment LED. The number is passed in the argument list of the function.
//
unsigned char Display(unsigned char no)
{
   unsigned char Pattern;
   unsigned char SEGMENT[] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,
                              0x7D,0x07,0x7F,0x6F};

   Pattern = SEGMENT[no];                          // Pattern to return
   return (Pattern);
}


//
// TMR0 timer interrupt service routine. The program jumps to the ISR at
// every 5ms.
//
void interrupt ()
{
      unsigned char Msd, Lsd;
      TMR0L = 100;                                 // Re-load TMR0
      INTCON = 0x20;                               // Set T0IE and clear T0IF
      Flag = ~ Flag;                               // Toggle Flag
      if(Flag == 0)                                // Do digit 1
      {
       DIGIT2 = 0;
       Msd = Cnt / 10;                             // MSD digit
       PORTC = Display(Msd);                       // Send to PORTC
       DIGIT1 = 1;                                 // Enable digit 1
      }
      else
      {                                            // Do digit 2
```

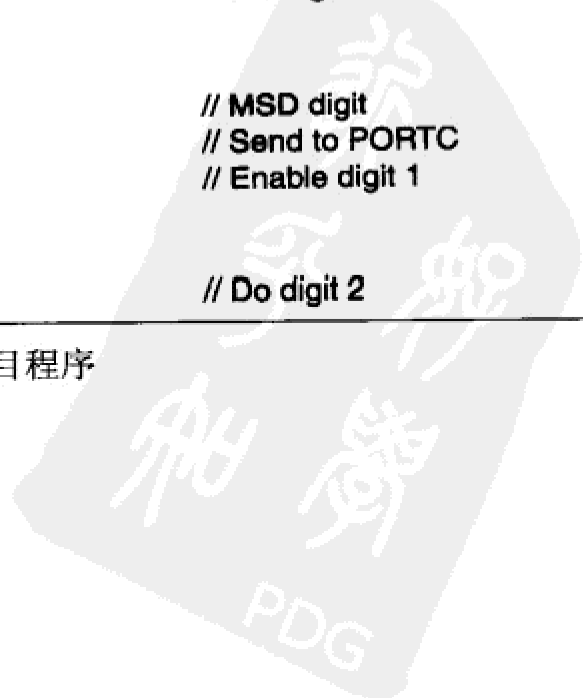


图6-36　项目程序

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
        DIGIT1 = 0;                                  // Disable digit 1
        Lsd = Cnt % 10;                              // LSD digit
        PORTC = Display(Lsd);                        // Send to PORTC
        DIGIT2 = 1;                                  // Enable digit 2
    }
}


//
// Start of MAIN Program. configure PORTB and PORTC as outputs.
// In addition, configure TMR0 to interrupt at every 10ms
//
void main()
{
    TRISC = 0;                                       // PORTC are outputs
    TRISB = 0;                                       // RB0, RB1 are outputs

    DIGIT1 = 0;                                      // Disable digit 1
    DIGIT2 = 0;                                      // Disable digit 2
//
// Configure TMR0 timer interrupt
//
    T0CON = 0xC4;                                    // Prescaler = 32
    TMR0L = 100;                                     // Load TMR0L with 100
    INTCON = 0xA0;                                   // Enable TMR0 interrupt
    Delay_ms(1000);

    for(;;)                                          // Endless loop
    {
      Cnt++;                                         // Increment Cnt
      if(Cnt == 100) Cnt = 0;                        // Count between 0 and 99
      Delay_ms(1000);                                // Wait 1 second
    }

}
```

图6-36　（续）

在中断服务子程序中，重新装载寄存器TMR0L，重新使能TMR0中断，将定时器中断标志清零以便生成下一个定时器中断。然后，交替地更新显示器的数位。一个叫作Flag的变量用来确定哪个数位需要更新。如项目6一样，调用函数Display来确定要发送至PORTC的位模式。

改进程序

329～331

在图6-36中，显示器的计数为00、01、…、09、10、11、…、99、00、01…（即小于10的数的第一个数位显示为0）。修改程序，使当需要显示的数小于10时，第一个数位不显示。修改过的程序（被命名为SEVEN5.C）如图6-37所示。这里，如果要显示0，则第一个数位（MSD）被禁止。

```
/*****************************************************
            Dual 7-SEGMENT DISPLAY COUNTER
            ==============================

In this project two common cathode 7-segment LED displays are
connected to PORTC of a PIC18F452 microcontroller and the
microcontroller is operated from a 4MHz resonator. Digit 1 (left
digit) enable pin is connected to port pin RB0 and digit 2
(right digit) enable pin is connected to port pin RB1 of the
microcontroller. The program counts up from 0 to 99 with one
second delay between each count.
```

图6-37　改进后的程序

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
The display is updated in a timer interrupt service routine at
every 5ms.

In this version of the program the first digit is blanked if the
number is 0.

Author:   Dogan Ibrahim
Date:     July 2007
File:     SEVEN5.C
***************************************************/
#define DIGIT1 PORTB.F0
#define DIGIT2 PORTB.F1

unsigned char Cnt = 0;
unsigned char Flag = 0;

//
// This function finds the bit pattern to be sent to the port to display a number
// on the 7-segment LED. The number is passed in the argument list of the function.
//
unsigned char Display(unsigned char no)
{
   unsigned char Pattern;
   unsigned char SEGMENT[] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,
                        0x7D,0x07,0x7F,0x6F};

   Pattern = SEGMENT[no];                          // Pattern to return
   return (Pattern);
}


//
// TMR0 timer interrupt service routine. The program jumps to the
// ISR at every 5ms.
//
void interrupt ()
{
      unsigned char Msd, Lsd;

  TMR0L = 100;                                // Re-load TMR0
  INTCON = 0x20;                              // Set T0IE and clear T0IF
  Flag = ~ Flag;                              // Toggle Flag
  if(Flag == 0)                               // Do digit 1
  {
   DIGIT2 = 0;
   Msd = Cnt / 10;                            // MSD digit
   if(Msd != 0)
   {
    PORTC = Display(Msd);                     // Send to PORTC
    DIGIT1 = 1;                               // Enable digit 1
   }
  }
  else
  {                                           // Do digit 2
   DIGIT1 = 0;                                // Disable digit 1
   Lsd = Cnt % 10;                            // LSD digit
   PORTC = Display(Lsd);                      // Send to PORTC
   DIGIT2 = 1;                                // Enable digit 2
  }
}
```

图6-37 （续）

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
//
// Start of MAIN Program. configure PORTB and PORTC as outputs.
// In addition, configure TMR0 to interrupt at every 10ms
//
void main()
{
    TRISC = 0;                                  // PORTC are outputs
    TRISB = 0;                                  // RB0, RB1 are outputs

    DIGIT1 = 0;                                 // Disable digit 1
    DIGIT2 = 0;                                 // Disable digit 2
//
// Configure TMR0 timer interrupt
//
    T0CON = 0xC4;                               // Prescaler = 32
    TMR0L = 100;                           // Load TMR0 with 100
    INTCON = 0xA0;                              // Enable TMR0 interrupt
    Delay_ms(1000);

    for(;;)                                     // Endless loop
    {
      Cnt++;                                    // Increment Cnt
      if(Cnt == 100) Cnt = 0;                   // Count between 0 and 99
      Delay_ms(1000);                           // Wait 1 second
    }

}
```

332~333

图6-37　（续）

项目6.8　带LCD显示器的伏特表

项目描述

在这个项目中，要设计一个带有LCD显示器的伏特表。伏特表可以被用来测量0~5V的电压。将需要测量的电压施加到PIC18F452型微控制器的一个模拟输入端。微控制器读取模拟电压值，并将其转换成数字，然后显示在LCD上。

在微控制器系统中，被测变量的输出通常使用LED、7段显示器或LCD显示器来显示。LCD可以显示字母数字或图像数据。有些LCD还具有40甚至更多的字符长度，可以显示多行。还有一些LCD可以用来显示图像。有些模块还提供彩色显示，有些模块还集成了背光功能，可以在比较昏暗的环境下显示。

就目前的接口技术而言，有两种基本类型的LCD：并行和串行。并行LCD（如日立的HD44780）通过至少一条数据线连接至微控制器，并且数据是以并行方式传送。无论是4条数据线还是8条数据线，它们都是共用的。一个4线的连接可以节省I/O引脚，但是速度变慢，因为数据要分两个阶段传送。串行LCD只通过一条数据线连接至微控制器，数据通常按照标准的RS-232异步数据通信协议传送至LCD。串行LCD更容易使用，但是花费也更多。

并行LCD的编程比较复杂，需要对LCD控制器的内部操作（包括时序图）有很好的理解。幸而mikroC语言提供特殊的库命令来显示字母数字和图像。用户所需要做的只是将LCD连接至微控制器，在软件中定义LCD的连接，然后发送特殊命令来将数据显示在LCD上。

HD44780 LCD模块

334

HD44780是当今最流行的一种字母数字LCD模块，深受工业界和业余爱好者们的推崇。该模块是单色的，具有不同的规格尺寸。有8、16、20、24、32和40列的模块可供选择。根据选择的不同模块，行数可能是1、2或4。显示器提供一个14引脚（或16引脚）的连接器，用来连接至微控制器。表6-10提供了14引脚LCD模块的引脚设置和引脚功能。下面是引脚功能的简要描述。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

表6-10 HD44780 LCD模块的引脚设置

引脚编号	名字	功能	引脚编号	名字	功能
1	V_{SS}	接地	8	D1	数据位1
2	V_{DD}	电源电压	9	D2	数据位2
3	V_{EE}	对比度	10	D3	数据位3
4	RS	选择寄存器	11	D4	数据位4
5	R/W	读/写	12	D5	数据位5
6	E	使能	13	D6	数据位6
7	D0	数据位0	14	D7	数据位7

Vss是0V电压或者接地。V_{DD}引脚应连接至电源的正极。尽管制造商指定了一个5 V的直流电源，但是该模块还是能够在低至3 V或者高至6 V的电压下正常工作。

引脚3称作V_{EE}，是对比度控制引脚。这个引脚用于调整显示器的对比度，应连接至一个可变的电压源。电位计通常连接至带有滑动端的电源线，而滑动端则连接到这个引脚，以便调整对比度。 335

引脚4是寄存器选择（RS），当这个引脚为低电压时，传输到显示器的数据被当作命令。当RS是高电平时，可以对模块读取或发送字符数据。

引脚5是读/写线。将这个引脚拉至低电平，以便向LCD模块中写入命令或字符数据。当这个引脚是高电平时，可以从模块中读出字符数据或状态信息。

引脚6是使能线，用于初始化模块和微控制器之间的命令或数据传送。当写入显示器时，数据只在该线的电平从高到低转换时传送。当从显示器读取数据时，只要在使能该引脚的电平从低至高转换后，数据才变得可用。并且，只要使能引脚一直保持逻辑高电平，数据就会一直有效。

引脚7到14是8位的数据总线（D0~D7)。数据在微控制器和LCD模块之间传送，可使用一个8位字节或两个4位的半字节。对于后者，只用到高四位的数据线（D4~D7)。4位模式意味着只有更少的4条I/O线用来与LCD通信。在这本书里，只使用基于字母数字的LCD和4位的接口。

连接LCD

在默认情况下，MikroC编译器假设LCD按如下方式连接至微控制器。

```
LCD     微控制器端口

D7      端口的第7位
D6      端口的第6位
D5      端口的第5位
D4      端口的第4位
E       端口的第3位
RS      端口的第2位
```

这里，port是使用`Lcd_Init`语句说明的端口名字。

例如，如果LCD以默认的连接方式连接到PORTB，那么可以使用`Lcd-Init(&PORTB)`语句。 336

以其他的方式连接LCD也是可能的，可以使用`Lcd_Config`命令来定义连接。

项目硬件

图6-38给出了项目的方框图。微控制器读取模拟电压，将其转换为数字，在格式化后，将其显示在LCD上。

该项目的电路原理图如图6-39所示。将待测的电压（0~5V之间） 施加到端口AN0，该端口已被软件设置为模拟输入。将LCD以默认的四线连接方式连接到微控制器的PORTC。使用一个电位计来调整LCD显示器的对比度。 337

项目PDL

该项目的PDL描述如图6-40所示。在程序的开始处，将PORTC 端口设定为输出，将PORTA端口设定为输入。然后，配置LCD和A/D转换器。接下来，程序进入一个无穷的循环，在循环中模拟输入电压被转换为数字，并被显示在LCD上。这个过程每秒重复一次。

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

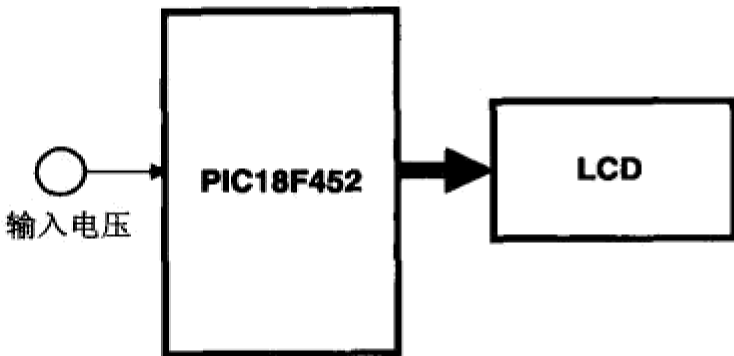


图6-38　项目的方框图

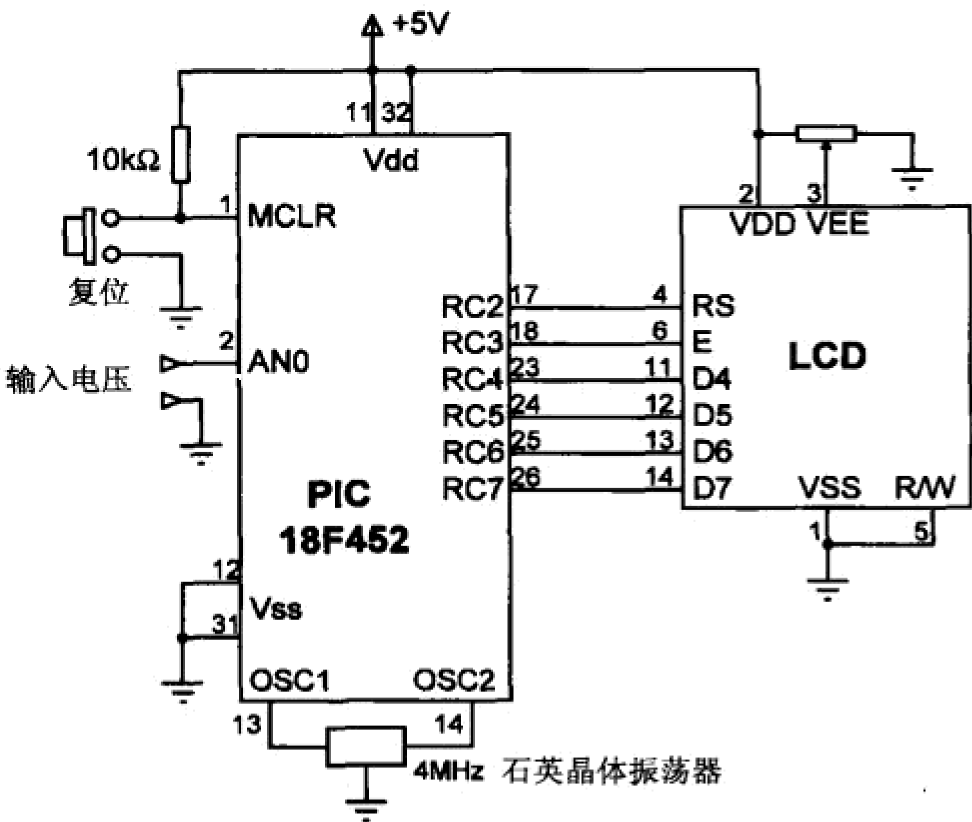
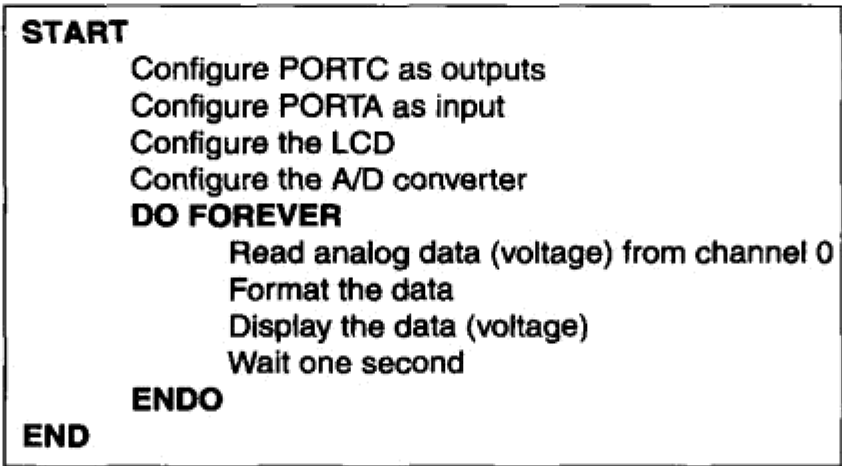


图6-39　项目的电路原理图



```
START
        Configure PORTC as outputs
        Configure PORTA as input
        Configure the LCD
        Configure the A/D converter
        DO FOREVER
                Read analog data (voltage) from channel 0
                Format the data
                Display the data (voltage)
                Wait one second
        ENDO
END
```

图6-40　项目的PDL

项目程序

将该程序命名为SEVEN6.C，程序清单如图6-41所示。在程序的开始处，将PORTC定义为输出，将PORTA定义为输入。然后设定LCD，将文本“VOLTMETER”显示在LCD上两秒钟。接着，配置A/D，将寄存器ADCON1设定为0x80，以便A/D转换的结果能向右对齐。将Vref电压设定为VDD(+5 V)，将所有PORTA引脚设定为模拟输入。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
/*****************************************************************
                    VOLTMETER WITH LCD DISPLAY
                    ==========================

In this project an LCD is connected to PORTC. Also, input port AN0 is used as
analog input. Voltage to be measured is applied to AN0. The microcontroller
reads the analog voltage, converts into digital, and then displays on the LCD.

Analog input range is 0 to 5V. A PIC18F452 type microcontroller is used in this
project, operated with a 4MHz resonator.

Analog data is read using the Adc_Read built-in function. This function uses the
internal RC clock for A/D timing.

The LCD is connected to the microcontroller as follows:

Microcontroller     LCD

   RC7              D7
   RC6              D6
   RC5              D5
   RC4              D4
   RC3              Enable
   RC2              RS

Author:     Dogan Ibrahim
Date:       July 2007
File:       SEVEN6.C
*****************************************************************/

//
// Start of MAIN Program. Configure LCD and A/D converter
//
void main()
{
    unsigned long Vin, mV;
    unsigned char op[12];
    unsigned char i,j,lcd[5];

    TRISC = 0;                              // PORTC are outputs (LCD)
    TRISA = 0xFF;                           // PORTA is input

//
// Configure LCD
//
    Lcd_Init(&PORTC);                       // LCD is connected to PORTC
    Lcd_Cmd(LCD_CLEAR);
    Lcd_Out(1,1,"VOLTMETER");
    Delay_ms(2000);
//
// Configure A/D converter. AN0 is used in this project
//
    ADCON1 = 0x80;                          // Use AN0 and Vref=+5V
//
// Program loop
//
    for(;;)                                 // Endless loop
    {
      Lcd_Cmd(LCD_CLEAR);
```

图6-41 项目清单

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
        Vin = Adc_Read(0);                       // Read from channel 0 (AN0)
        Lcd_Out(1,1,"mV = ");                    // Display "mV = "
        mV = (Vin * 5000) >> 10;                 // mv = Vin x 5000 / 1024
        LongToStr(mV,op);                        // Convert to string in "op"
//
// Remove leading blanks
//
        j=0;
        for(i=0;i<=11;i++)
        {
          if(op[i] != ' ')                       // If a blank
          {
            lcd[j]=op[i];
            j++;
          }
        }
//
// Display result on LCD
//
        Lcd_Out(1,6,lcd);                        // Output to LCD
        Delay_ms(1000);                          // Wait 1 second
    }

}
```

图6-41　（续）

338~339

主程序以一个for 语句开始循环。在循环中LCD被清零，使用Adc_Read(0)语句从通道0（AN0引脚）读入模拟数据。将转换后的数字数据存储在unsingned long类型的变量Vin中。A/D是10位宽度，因此对应于5 000 mV的参考电压，有1 024步（0~1 023）。每一步相当于5 000 mV/1 024=4.8 mV。在循环中，将变量Vin乘以5 000，再除以1 024，可以转换成毫伏。向右移10位，完成除法。此时，变量mV包含了以毫伏表示的转换数据。

调用函数LongToStr，将mV转换为一个字符串，形成字符数组op。函数LongToSt将一个长整型变量转换为一个固定宽度为11个字符的字符串。如果所得的字符串少于11个字符，那么数据的左边就以空白字符填充。

340

删除最前面的空白，将数据存储到一个叫作lcd的变量。调用函数Lcd_Out，将数据显示在LCD上，从第一行的第5列开始显示数据。例如，如果被测电压是1 267 mV，那么在LCD上显示的将是：

mV=1267

一种更精确的显示

图6-41中显示的电压不是非常精确，因为在计算中执行的是整数计算，将A/D输出乘以5 000，然后以整数除法除以1 024，得到电压值。尽管乘法是精确的，可是当结果被除以1 024的时候精确度就下降了。更精确的结果可以通过在显示之前缩放数的方法来实现，其方法如下。

首先，将数Vin乘以一个因数来避免整数除法。例如，因为5 000/1 024=4.88，可以将Vin乘以488。为了显示，可以将结果除以100，得到整数部分，将小数部分作为余数。整数部分和小数部分以小数点连接。该技术已在程序SEVEN7.C中实现，如图6-42所示。在这个程序中，变量Vdec和Vfrac分别用来存储整数和小数部分。然后使用函数LongToStr将小数部分转化为字符串，并且将开头的空白去掉。小数部分被称为ch1和ch2。将这些数加上48，可以得到转换的字符，然后使用LCD命令Lcd_Chr_Cp将它们显示在下一个光标位置。

也可以使用浮点数来计算和显示更为精确的结果，但是这样会占用大量的内存，所以还是应该尽量避免使用这种方法。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
/*******************************************************************
            VOLTMETER WITH LCD DISPLAY
            ============================

In this project an LCD is connected to PORTC. Also, input port
AN0 is used as analog input. Voltage to be measured is applied
to AN0. The microcontroller reads the analog voltage, converts
into digital, and then displays on the LCD.

Analog input range is 0 to 5V. A PIC18F452 type microcontroller
is used in this project, operated with a 4MHz resonator.

Analog data is read using the Adc_Read built-in function. This
function uses the internal RC clock for A/D timing.

The LCD is connected to the microcontroller as follows:

Microcontroller     LCD

    RC7             D7
    RC6             D6
    RC5             D5
    RC4             D4
    RC3             Enable
    RC2             RS

This program displays more accurate results than program SEVEN6.C.
The voltage is displayed as follows:

    mV = nnnn.mm

Author:     Dogán Ibrahim
Date:       July 2007
File:       SEVEN7.C
*******************************************************************/

//
// Start of MAIN Program. Configure LCD and A/D converter
//
void main()
{
    unsigned long Vin, mV,Vdec,Vfrac;
    unsigned char op[12];
    unsigned char i,j,lcd[5],ch1,ch2;

    TRISC = 0;                              // PORTC are outputs (LCD)
    TRISA = 0xFF;                           // PORTA is input

//
// Configure LCD
//
    Lcd_Init(&PORTC);                       // LCD is connected to PORTC
    Lcd_Cmd(LCD_CLEAR);
    Lcd_Out(1,1,"VOLTMETER");
    Delay_ms(2000);
//
// Configure A/D converter. AN0 is used in this project
//
    ADCON1 = 0x80;                          // Use AN0 and Vref=+5V
```

图6-42　一个更精确的程序

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源


```
//
// Program loop
//
   for(;;)                              // Endless loop
   {
     Lcd_Cmd(LCD_CLEAR);
     Vin = Adc_Read(0);                 // Read from channel 0 (AN0)
     Lcd_Out(1,1,"mV = ");              // Display "mV = "
     Vin = 488*Vin;                     // Scale up the result
     Vdec = Vin / 100;                  // Decimal part
     Vfrac = Vin % 100;                 // Fractional part
     LongToStr(Vdec,op);                // Convert Vdec to string in "op"
//
// Remove leading blanks
//
     j=0;
     for(i=0;i<=11;i++)
     {
      if(op[i] != ' ')                  // If a blank
      {
        lcd[j]=op[i];
        j++;
      }
     }
//
// Display result on LCD
//
     Lcd_Out(1,6,lcd);                  // Output to LCD
     Lcd_Out_Cp(".");                   // Display "."
     ch1 = Vfrac / 10;                  // Calculate fractional part
     ch2 = Vfrac % 10;                  // Calculate fractional part
     Lcd_Chr_Cp(48+ch1);                // Display fractional part
     Lcd_Chr_Cp(48+ch2);                // Display fractional part
     Delay_ms(1000);                    // Wait 1 second
   }

}
```

图6-42 （续）

项目6.9 带键盘和LCD的计算器

项目描述

341～343

键盘是用来向微控制器系统输入数字型或字母数字型数据的。键盘有很多不同的种类和规格可供选择，从2×2到4×4，甚至更大。

本项目使用一个4×4键盘（如图6-43所示）和一个LCD来设计一个简单的计算器。

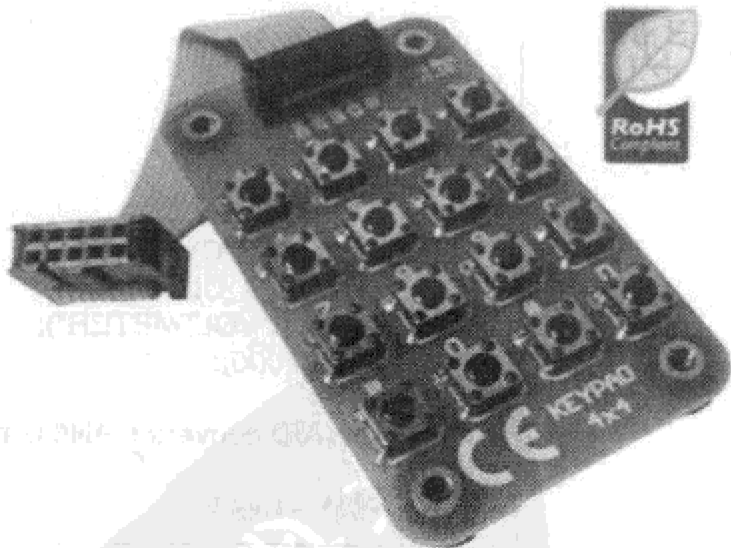


图6-43 4×4键盘

图6-44给出了项目中所使用的键盘结构，它由4×4阵列形式的16个开关组成，这些开关分别对应着数字0~9、ENTER键、“+”、“.”、“−”、“*”和“/”。将键盘的行和列连接至一个微控制器的PORTB端口，微控制器会通过扫描键盘来探测开关是否按下。键盘的操作过程如下。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

■ 将逻辑1通过RB0施加到第一列。

■ 读取端口引脚RB4~RB7的状态。如果数据是非零的，那么表明有一个开关被按下。若RB4为1，则键1被按下；若RB5为1，则键4被按下；若RB6为1，则键9被按下；依此类推。

■ 将逻辑1通过RB1施加到第二列。

■ 同样地，读取端口引脚RB4~RB7的状态。如果数据是非零的，那么表明有一个开关被按下。若RB4为1，则键2被按下；若RB5为1，键6被按下；若RB6为1，则键0被按下；依此类推。

■ 重复以上的过程，直到所有4列全部被扫描完毕。

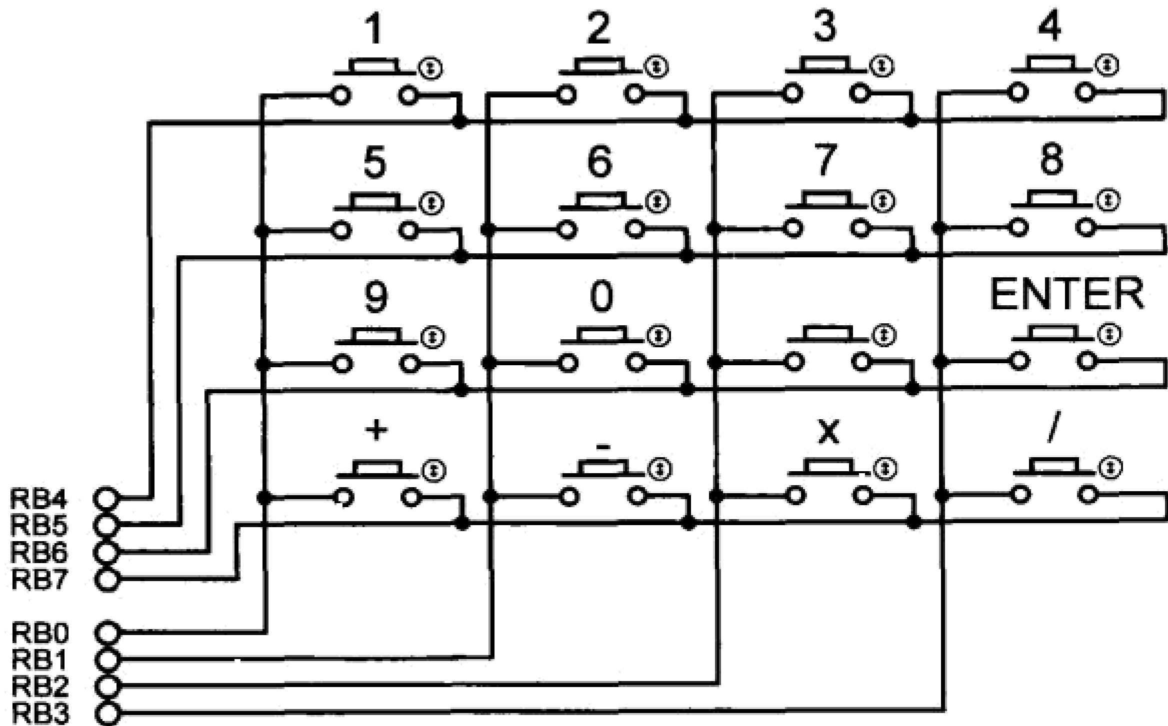


图6-44　4×4键盘结构

这个项目设计了一个简单的整数计算器。计算器可以执行整数的加法、减法、乘法和除法运算，并且在LCD上显示计算结果。计算器的操作过程如下：当将电源施加在系统上时，LCD显示器显示文本“CALCULATOR”两秒钟；然后在LCD的第一行显示文本“No.1:”，等待用户键入第一个数字，然后按ENTER键确认；接下来，在LCD的第二行显示文本“No.2:”，等待用户键入第二个数字，并按ENTER键确认；在这之后，按下合适的运算操作键；所得的计算结果将在LCD上显示5秒钟，然后被清除，为下一次计算做好准备。下面的例子演示了如何将数12和20相加： 344

```
No1: 12   ENTER
No2:20    ENTER
Op:+
Res=32
```

在该项目中，键盘的按键功能标注为：

```
1    2    3    4
5    6    7    8
9    0         ENTER
+    -    ×    /
```

在0与ENTER键之间的那个键，在本项目中没有使用到。 345

项目硬件

该项目的方框图如图6-45所示。其电路原理图如图6-46所示。该项目使用了一个PIC18F452型微控制器，工作时钟来自4MHz的晶体振荡器。键盘的列连接至微控制器的端口引脚RB0~RB3，而键盘的行通过下拉电阻连接至端口引脚RB4~RB7。LCD以默认模式连接至PORTC端口，如图6-39所示。另外，还设置一个外部复位按钮来在必要时复位微控制器。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

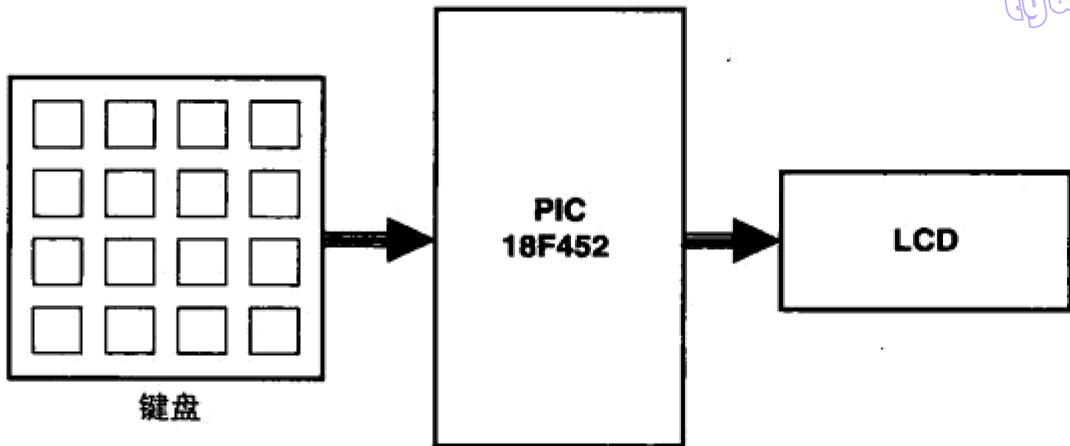


图6-45　项目的方框图

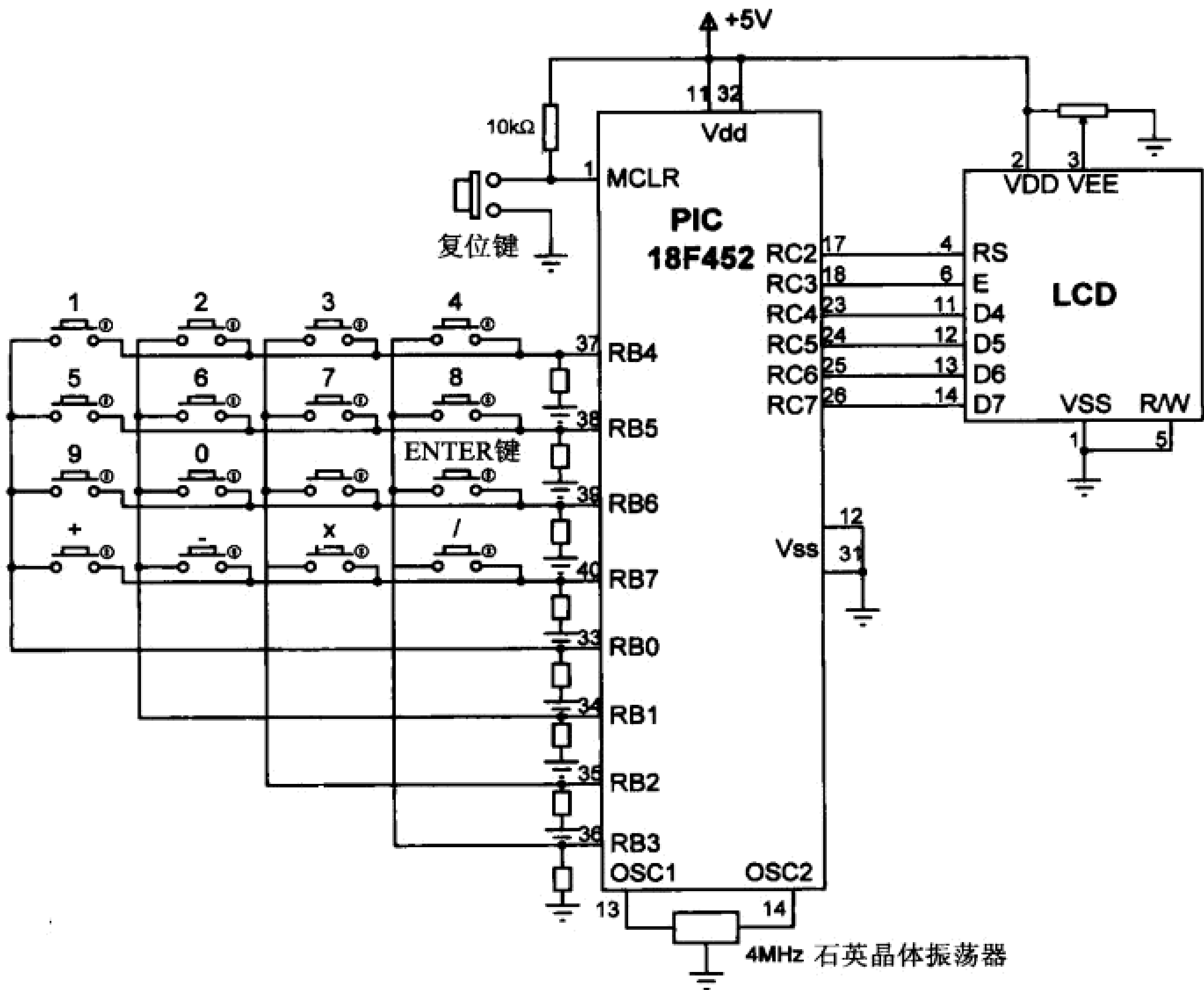


图6-46　项目的电路原理图

项目PDL

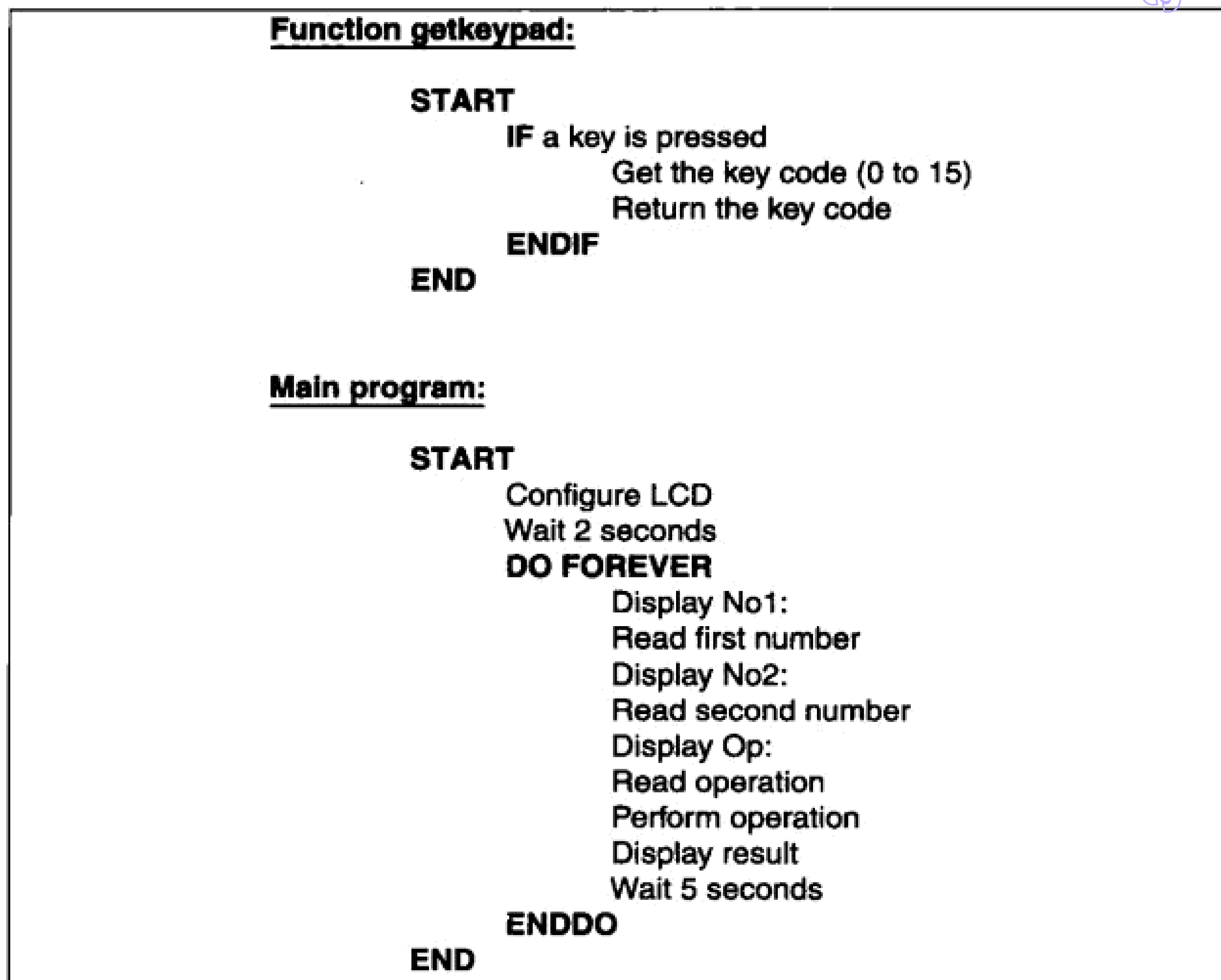
该项目的PDL描述如图6-47所示。项目程序由两个部分组成：函数getkeypad和主程序。函数getkeypad从键盘接收一个按键。在主程序中，从键盘接收两个数和所需的运算操作。微控制器执行所需的运算操作，然后将结果显示在LCD上。

项目程序

程序KEYPAD.C的程序清单如图6-48所示。每个键都被分配一个数值，如下：

```
0    1    2    3
4    5    6    7
8    9    10   11
12   13   14   15
```

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源



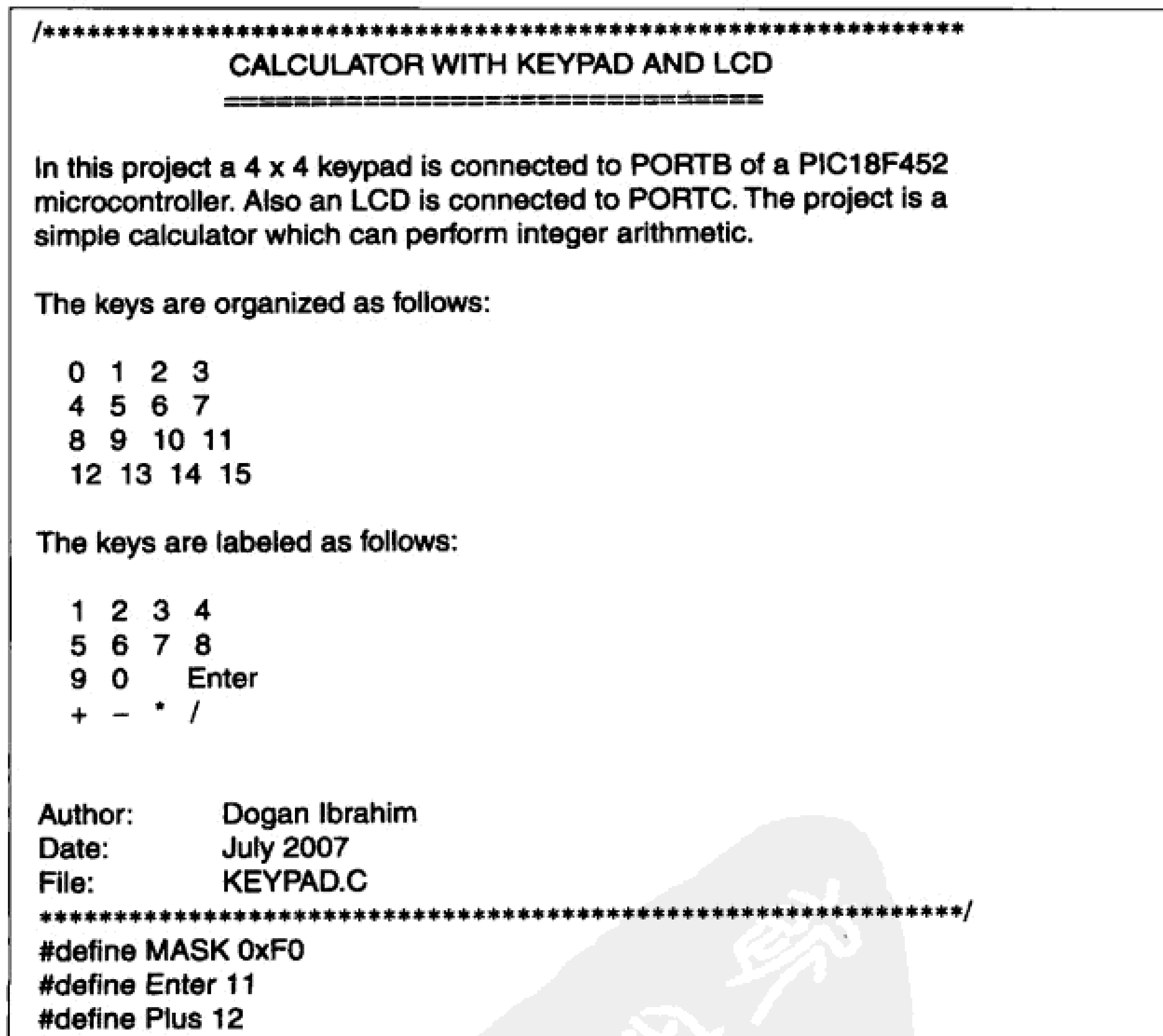
```
Function getkeypad:

        START
              IF a key is pressed
                     Get the key code (0 to 15)
                     Return the key code
              ENDIF
        END


Main program:

        START
              Configure LCD
              Wait 2 seconds
              DO FOREVER
                     Display No1:
                     Read first number
                     Display No2:
                     Read second number
                     Display Op:
                     Read operation
                     Perform operation
                     Display result
                     Wait 5 seconds
              ENDDO
        END
```

图6-47 项目的PDL



```
/*****************************************************************
              CALCULATOR WITH KEYPAD AND LCD
              ==============================

In this project a 4 x 4 keypad is connected to PORTB of a PIC18F452
microcontroller. Also an LCD is connected to PORTC. The project is a
simple calculator which can perform integer arithmetic.

The keys are organized as follows:

   0  1  2  3
   4  5  6  7
   8  9  10 11
   12 13 14 15

The keys are labeled as follows:

   1  2  3  4
   5  6  7  8
   9  0     Enter
   +  -  *  /


Author:      Dogan Ibrahim
Date:        July 2007
File:        KEYPAD.C
*****************************************************************/
#define MASK 0xF0
#define Enter 11
#define Plus 12
```

图6-48 程序清单

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源


```
#define Minus 13
#define Multiply 14
#define Divide 15

//
// This function gets a key from the keypad
//
unsigned char getkeypad()
{
      unsigned char i, Key = 0;

      PORTB = 0x01;                                   // Start with column 1
      while((PORTB & MASK) == 0)                      // While no key pressed
      {
         PORTB = (PORTB << 1);                        // next column
         Key++;                                       // column number
         if(Key == 4)
         {
            PORTB = 0x01;                             // Back to column 1
            Key = 0;
         }
      }
      Delay_Ms(20);                                   // Switch debounce

      for(i = 0x10; i !=0; i <<=1)                    // Find the key pressed
      {
         if((PORTB & i) != 0)break;
         Key = Key + 4;
      }

      PORTB=0x0F;
      while((PORTB & MASK) != 0);                     // Wait until key released
      Delay_Ms(20);                                   // Switch debounce

      return (Key);                                   // Return key number
}


//
// Start of MAIN program
//
void main()
{
     unsigned char MyKey, i,j,lcd[5],op[12];
     unsigned long Calc, Op1, Op2;

     TRISC = 0;                              // PORTC are outputs (LCD)
     TRISB = 0xF0;                           // RB4-RB7 are inputs

//
// Configure LCD
//
       Lcd_Init(&PORTC);                     // LCD is connected to PORTC
       Lcd_Cmd(LCD_CLEAR);
       Lcd_Out(1,1,"CALCULATOR");            // Display CALCULATOR
       Delay_ms(2000);                       // Wait 2 seconds
       Lcd_Cmd(LCD_CLEAR);                   // Clear display
//
```

图6-48 （续）

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
// Program loop
//
    for(;;)                                    // Endless loop
    {
      MyKey = 0;
      Op1 = 0;
      Op2 = 0;

      Lcd_Out(1,1,"No1: ");                    // Display No1:
      while(1)                                 // Get first no
      {
        MyKey = getkeypad();
        if(MyKey == Enter)break;               // If ENTER pressed
        MyKey++;
        if(MyKey == 10)MyKey = 0;              // If 0 key pressed
        Lcd_Chr_Cp(MyKey + '0');
        Op1 = 10*Op1 + MyKey;                  // First number in Op1
      }

      Lcd_Out(2,1,"No2: ");                    // Display No2:
      while(1)                                 // Get second no
      {
        MyKey = getkeypad();
        if(MyKey == Enter)break;               // If ENTER pressed
        MyKey++;
        if(MyKey == 10)MyKey = 0;              // If 0 key pressed
        Lcd_Chr_Cp(MyKey + '0');
        Op2 = 10*Op2 + MyKey;                  // Second number in Op2
      }

      Lcd_Cmd(LCD_CLEAR);                      // Clear LCD
      Lcd_Out(1,1,"Op: ");                     // Display Op:

      MyKey = getkeypad();                     // Get operation
      Lcd_Cmd(LCD_CLEAR);
      Lcd_Out(1,1,"Res=");                     // Display Res=
      switch(MyKey)                            // Perform the operation
      {
        case Plus:
             Calc = Op1 + Op2;                 // If ADD
             break;
        case Minus:
             Calc = Op1 - Op2;                 // If Subtract
             break;
        case Multiply:
             Calc = Op1 * Op2;                 // If Multiply
             break;
        case Divide:
             Calc = Op1 / Op2;                 // If Divide
             break;
      }

      LongToStr(Calc, op);                     // Convert to string in op
//
// Remove leading blanks
//
      j=0;
      for(i=0;i<=11;i++)
      {
        if(op[i] != ' ')                       // If a blank
```

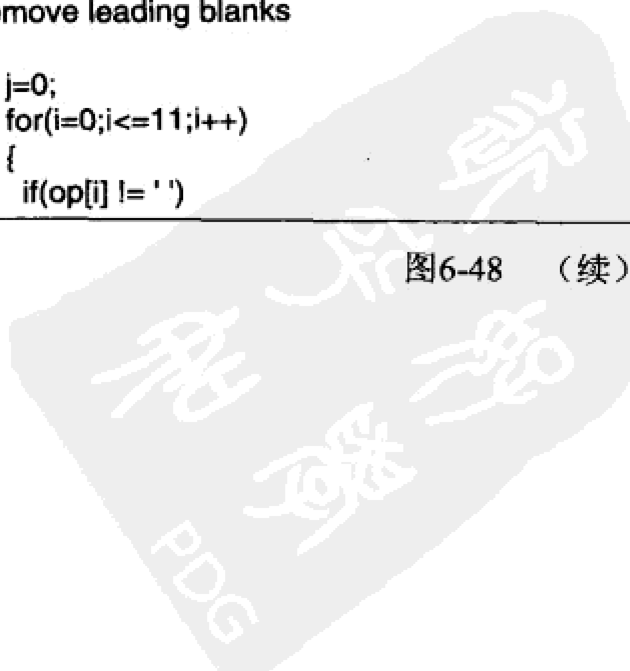


图6-48 （续）

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com


```
            {
                lcd[j]=op[i];
                j++;
            }
          }

          Lcd_Out_Cp(lcd);                          // Display result
          Delay_ms(5000);                           // Wait 5 seconds
          Lcd_Cmd(LCD_CLEAR);
      }
  }
```

图6-48　（续）

程序由getkeypad函数和主程序组成，函数getkeypad用来读入按下的键。变量MyKey用来存储按下的键值（0~15），变量Op1和Op2分别用来存储用户键入的第一个数和第二个数。所有这些变量在程序开始时都被清零。while循环用来读入第一个数，然后存储在变量Op1中。当用户按下ENTER键时，退出本次循环。同样地，在第二个while循环，从键盘读入第二个数。然后读入要执行的运算操作，并存储到变量MyKey。switch语句用来执行所需要的运算操作，并将运算结果存储到变量Calc。函数LongToStr用来将运算结果转化为字符串数组。前面的空白字符将像在项目8中一样被去除。程序将结果显示在LCD上，等待5秒钟后，清除屏幕等待下一次计算。无限重复这个过程。

347

函数getkeypad从键盘接收一个键。向第一列发送逻辑1，然后检查所有的行。当有按键被按下时，在相应的行将探测出逻辑1，程序跳出while循环。接下来，使用一个for循环来查找用户实际按下的键，并记为一个0~15之间的数字。

很重要的一点是，当一个键被按下或松开时，存在接触噪声信号，即键盘输出表现为瞬时的上升和下降脉冲，在输出段产生大量的逻辑0和逻辑1脉冲。开关接触噪声通常可以用硬件消除，或者在接触防反跳过程中进行编程加以消除。在软件方法中，消除接触噪声的最简单方法是在开关键被按下或释放后等待20ms左右。在图6-46中，接触防反跳功能在函数getkeypad中实现。

348~351

使用内置键盘函数的程序

在图6-48所示的程序清单中，函数getkeypad用来从键盘读入一个键。mikroC语言带有内置函数keypad_Read和Keypad_Released，可分别在按键被按下和释放时从键盘读入一个键。图6-49给出了修正过的程序清单（KEYPAD2.C），这里使用内置函数Keypad_Released来实现前述的计算器项目。其电路原理图和图6-46所示的一样。

```
/*******************************************************************
                 CALCULATOR WITH KEYPAD AND LCD
                 ==============================

In this project a 4 x 4 keypad is connected to PORTB of a PIC18F452
microcontroller. Also an LCD is connected to PORTC. The project is a simple
calculator which can perform integer arithmetic.

The keys are labeled as follows:

    1  2  3  4
    5  6  7  8
    9  0     Enter
    +  -  *  /


In this program mikroC built-in functions are used.
```

图6-49　修改后的程序清单

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源


```
Author:     Dogan Ibrahim
Date:       July 2007
File:       KEYPAD2.C
*******************************************************************************/

#define Enter 12
#define Plus 13
#define Minus 14
#define Multiply 15
#define Divide 16

//
// Start of MAIN program
//
void main()
{
   unsigned char MyKey, i,j,lcd[5],op[12];
   unsigned long Calc, Op1, Op2;

   TRISC = 0;                              // PORTC are outputs (LCD)
//
// Configure LCD
//
     Lcd_Init(&PORTC);                     // LCD is connected to PORTC
     Lcd_Cmd(LCD_CLEAR);
     Lcd_Out(1,1,"CALCULATOR");            // Display CALCULATOR
     Delay_ms(2000);
     Lcd_Cmd(LCD_CLEAR);
//
// Configure KEYPAD
//
   Keypad_Init(&PORTB);                    // Keypad on PORTB

//
// Program loop
//
   for(;;)                                         // Endless loop
   {
     MyKey = 0;
     Op1 = 0;
     Op2 = 0;

     Lcd_Out(1,1,"No1: ");                         // Display No1:
     while(1)
     {
       do                                          // Get first number
          MyKey = Keypad_Released();
       while(!MyKey);
       if(MyKey == Enter)break;                    // If ENTER pressed
       if(MyKey == 10)MyKey = 0;                   // If 0 key pressed
       Lcd_Chr_Cp(MyKey + '0');
       Op1 = 10*Op1 + MyKey;
     }

     Lcd_Out(2,1,"No2: ");                         // Display No2:
     while(1)                                      // Get second no
     {
       do
          MyKey = Keypad_Released();               // Get second number
       while(!MyKey);
```

图6-49 （续）

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
            if(MyKey == Enter)break;                    // If ENTER pressed
            if(MyKey == 10)MyKey = 0;                   // If 0 key pressed
            Lcd_Chr_Cp(MyKey + '0');
            Op2 = 10*Op2 + MyKey;
        }

        Lcd_Cmd(LCD_CLEAR);
        Lcd_Out(1,1,"Op: ");                            // Display Op:

        do
          MyKey = Keypad_Released();                    // Get operation
        while(!MyKey);
        Lcd_Cmd(LCD_CLEAR);
        Lcd_Out(1,1,"Res=");                            // Display Res=
        switch(MyKey)                                   // Perform the operation
        {
         case Plus:
              Calc = Op1 + Op2;                         // If ADD
              break;
         case Minus:
              Calc = Op1 - Op2;                         // If Subtract
              break;
         case Multiply:
              Calc = Op1 * Op2;                         // If Multiply
              break;
         case Divide:
              Calc = Op1 / Op2;                         // If Divide
              break;
        }

        LongToStr(Calc, op);                            // Convert to string
//
// Remove leading blanks
//
        j=0;
        for(i=0;i<=11;i++)
        {
         if(op[i] != ' ')                               // If a blank
         {
           lcd[j]=op[i];
           j++;
         }
        }

        Lcd_Out_Cp(lcd);                                // Display result
        Delay_ms(5000);                                 // Wait 5 seconds
        Lcd_Cmd(LCD_CLEAR);
      }
}
```

图6-49 （续）

在使用函数Keypad_Released之前，必须调用函数Keypad_Init来指明键盘所连接的微控制器。函数Keypad_Released可以探测出一个键被按下然后被释放的情况。当按键被释放时，函数返回一个对应于被释放键在1~16之间的数。程序的其余部分和图6-48中一样。

项目 6.10 基于串行通信的计算器

项目描述

串行通信是一种快速可靠地远距离发送数据的简单方式。最常见的串行通信方法是基于RS232标准

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

的，标准数据通过一条线路按照位串行格式以指定的速率从发送设备传输到接收设备，所给定的速率也称作波特率，或每秒钟发送的比特数。常见的波特率有4 800、9 600、19 200、38 400，等等。

RS232串行通信是一种异步数据传送模式，在这里数据是一个字符一个字符地传送的。每个字符包括一个开始位、7或8位数据位、一个可选的奇偶位和一个或多个停止位。最常见的格式是8个数据位、无奇偶位和一个停止位。最低有效位数据最先被传送，而最高有效位则最后被传送。 352~354

这里，逻辑1（高电平）被定义为−12 V，而逻辑0（低电平）被定义为+12 V。图6-50给出了字符A（ASCII的二进制模式为00100001）是如何通过一条串行线路发送的。在空闲状态时，线路通常是−12 V。开始位在数据线从高电平变为低电平时首先被发送。然后发送8个数据位，从最低有效位开始发送。最后，在数据线从由低电平到高电平时发送停止位。

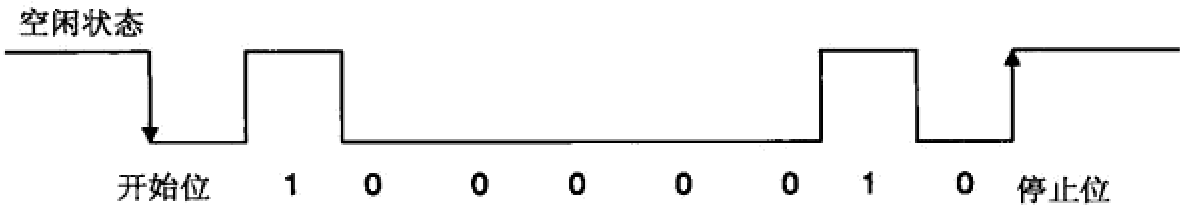


图6-50 以串行格式发送字符A 355

在串行连接中，通信最少需要3条线路：发送（TX）、接收（RX）和接地（GND）。串行设备使用两种类型的连接器进行连接：9引脚和25引脚。表6-11给出了每种类型连接器的TX、RX和接地引脚。在RS232串行通信中使用的连接器如图6-51所示。

表6-11 串行通信需要的最少引脚

9引脚的连接器		25引脚的连接器	
引 脚	功 能	引 脚	功 能
2	发送（TX）	2	发送（TX）
3	接收（RX）	3	接收（RX）
5	接地（GND）	5	接地（GND）

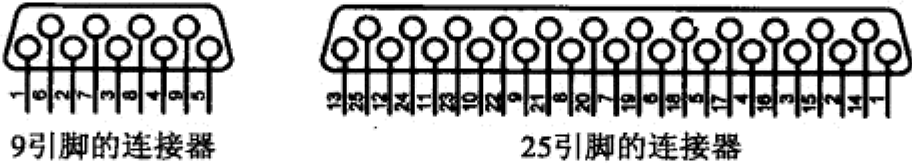


图6-51 RS232连接器

如前所述，RS232电压范围为±12 V。然而，因为微控制器输入输出端口的电压范围为0~+5 V，所以在将微控制器连接至RS232兼容设备前，必须先将电压进行转换。因此在将RS232兼容设备连接至微控制器之前，必须将微控制器的输出信号转化为±12 V，而将RS232设备输入到微控制器的信号转化为0~+5 V。通常使用特殊的RS232电压转换芯片来完成电压转换。比较常用的芯片是MAX232，它是一个两路的转换芯片，其引脚配置如图6-52所示。该芯片需要外接4个1 μF的电容才能正常工作。 356

在PIC18系列微控制器中，串行通信可以硬件或者软件方式实现。硬件方式较容易，PIC18微控制器带有内置的USART（通用的同步异步接收发送器）电路，用来为串行通信提供特殊的输入/输出引脚。对于串行通信，所有数据的传送都是由USART来处理的，但是在发送和接收数据之前必须对USART进行配置。在软件方式下，所有的串行位时序都在软件中实现，任何输入/输出引脚都可以被编程用于串行通信。

在这个项目中，将PC通过RS232电缆连接至微控制器。项目实现了一个简单的整数计算器，其中所需的数据使用PC键盘输入，然后被串行地发送至微控制器，并在PC显示器上进行显示。 357

下面是一个简单的计算示例：

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
CALCULATOR PROGRAM

Enter First Number: 12
Enter Second Number: 2
Enter Operation: +
Result = 14
```

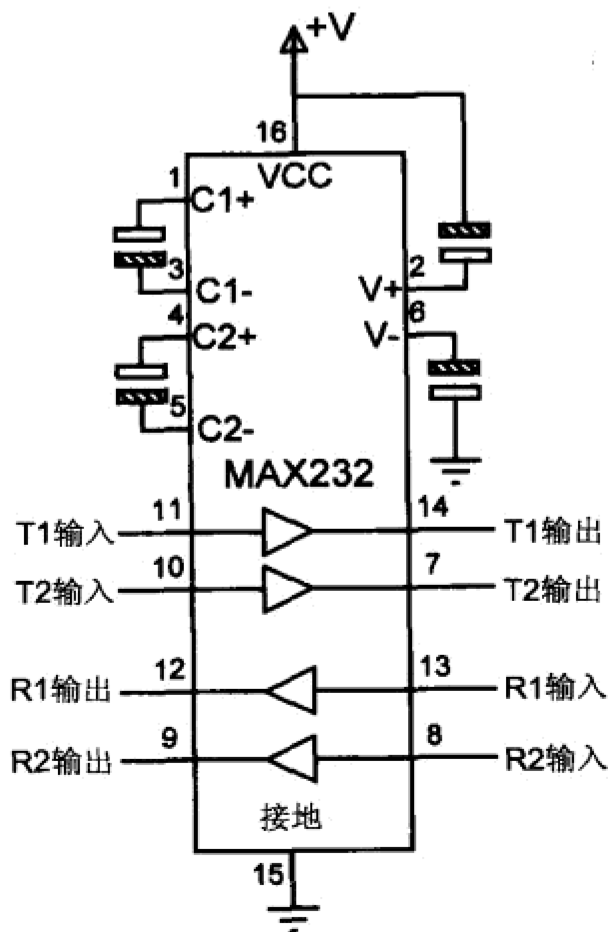


图6-52　MAX引脚配置

项目硬件

图6-53给出了项目的方框图。其电路原理图如图6-54所示。项目使用PIC18F452型微控制器，晶体振荡器时钟为4 MHz，使用内置的USART来进行串行通信。将微控制器的串行通信线（RC6和RC7）连接到MAX232电压转换芯片，然后再使用9引脚的连接器连接至PC的串行输入端口（COM1）。

项目PDL

该项目的PDL描述如图6-55所示。项目由一个主程序和两个名为Newline和Text_To_User的函数组成。函数Newline用来向串行端口发送载波和线馈信号。函数Text_To_User用来向USART发送文本信息。主程序从PC键盘接收两个数和需要执行的运算操作。接收的数字将被回显在PC显示器上。运算所得的结果也将在显示器上显示。

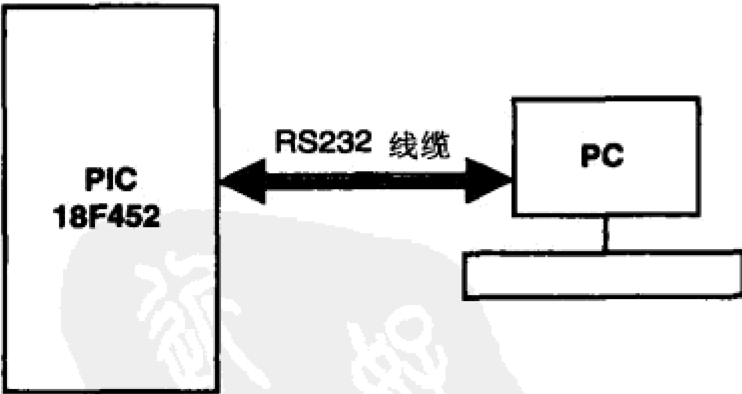


图6-53　项目的方框图

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

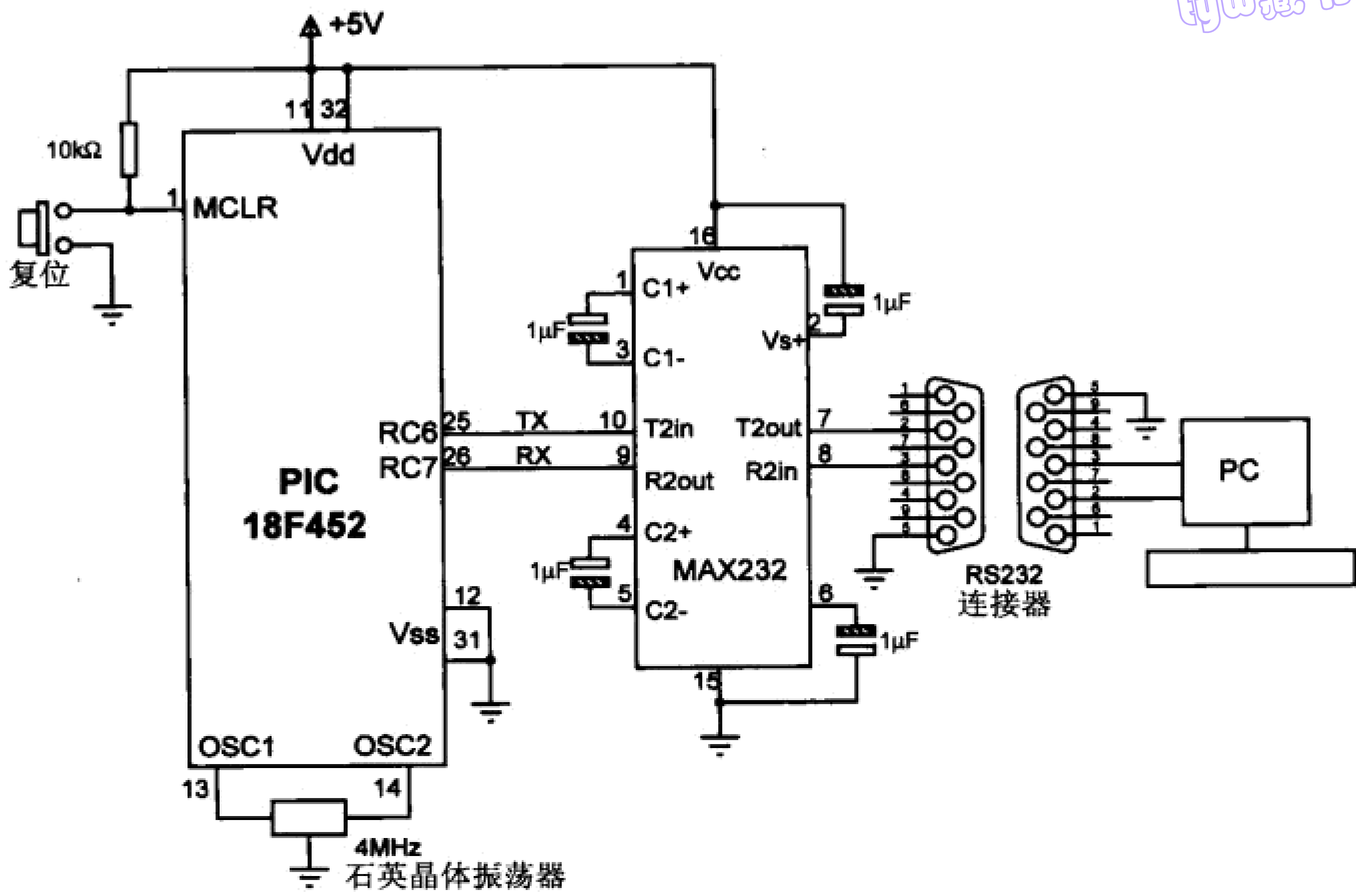
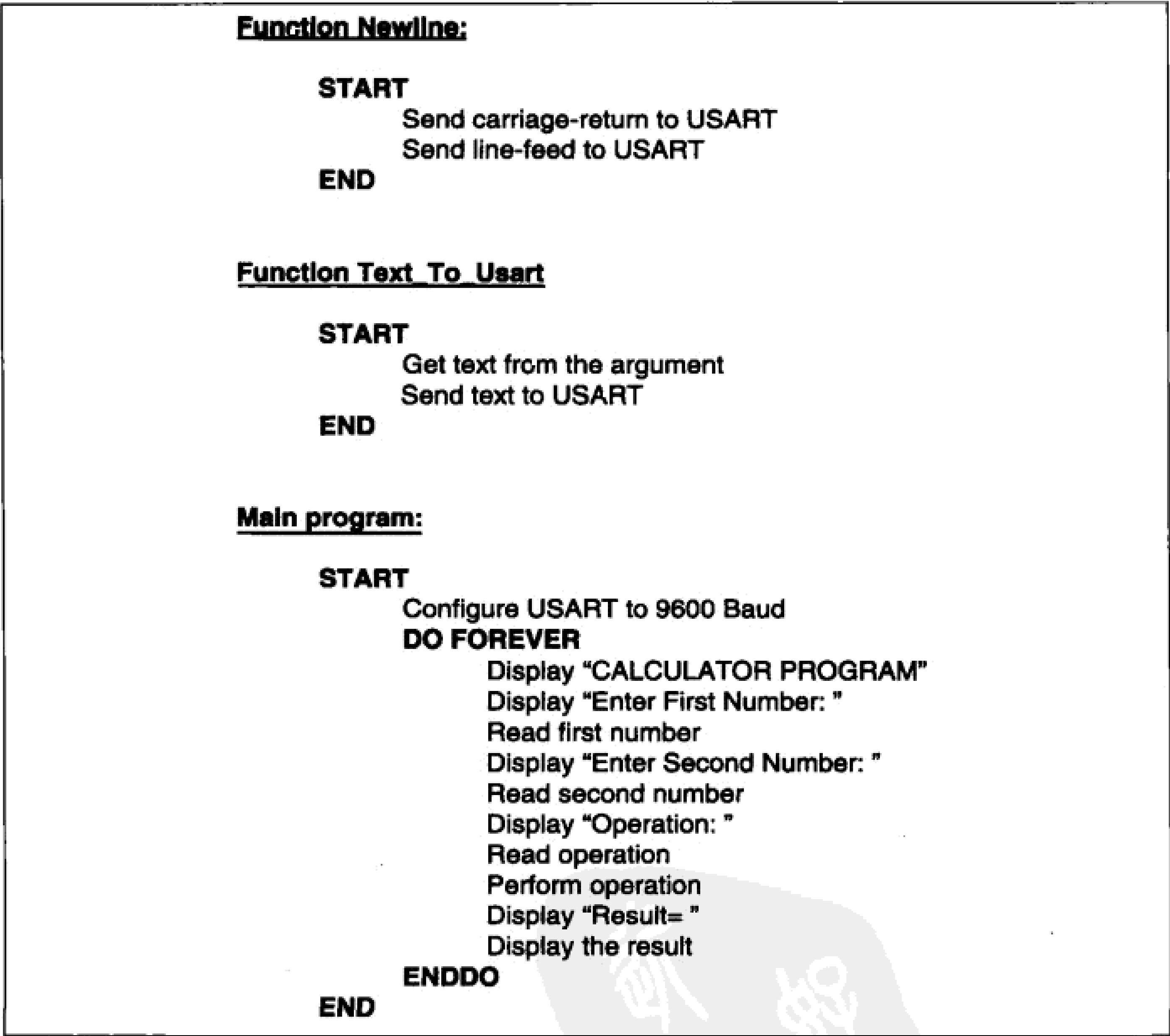


图6-54 项目的电路原理图



```
Function Newline:

    START
        Send carriage-return to USART
        Send line-feed to USART
    END

Function Text_To_Usart

    START
        Get text from the argument
        Send text to USART
    END

Main program:

    START
        Configure USART to 9600 Baud
        DO FOREVER
            Display "CALCULATOR PROGRAM"
            Display "Enter First Number: "
            Read first number
            Display "Enter Second Number: "
            Read second number
            Display "Operation: "
            Read operation
            Perform operation
            Display "Result= "
            Display the result
        ENDDO
    END
```

图6-55 项目的PDL

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

项目程序

该项目的程序清单如图6-56所示。程序包括一个主程序和两个名为Newline和Text_To_User的函数。函数Newline用来向USART发送载波和线馈信号，以将光标移动到下一行。函数Text_To_User用来向USART发送文本信息。

```
/*************************************************************************
                    CALCULATOR WITH PC INTERFACE
                    ==============================

In this project a PC is connected to a PIC18F452 microcontroller. The project is a
simple integer calculator. User enters the numbers using the PC keyboard. Results are
displayed on the PC monitor.

The following operations can be performed:

  + - * /

This program uses the built in USART of the microcontroller. The USART is
configured to operate with 9600 Baud rate.

The serial TX pin is RC6 and the serial RX pin is RC7.

Author:     Dogan Ibrahim
Date:       July 2007
File:       SERIAL1.C
*************************************************************************/

#define Enter 13
#define Plus '+'
#define Minus '-'
#define Multiply '*'
#define Divide '/'

//
// This function sends carriage-return and line-feed to USART
//
void Newline()
{
  Usart_Write(0x0D);                          // Send carriage-return
  Usart_Write(0x0A);                          // Send line-feed
}


//
// This function sends a text to USART
//
void Text_To_Usart(unsigned char *m)
{
  unsigned char i;

  i = 0;
  while(m[i] != 0)
  {                                           // Send TEXT to USART
    Usart_Write(m[i]);
    i++;
  }
}
```

图6-56　程序清单

PDG

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源


```
//
// Start of MAIN program
//
void main()
{
    unsigned char MyKey, i,j,kbd[5],op[12];
    unsigned long Calc, Op1, Op2,Key;
    unsigned char msg1[] = "    CALCULATOR PROGRAM";
    unsigned char msg2[] = "     Enter First Number: ";
    unsigned char msg3[]= "Enter Second Nummber: ";
    unsigned char msg4[] = "          Enter Operation: ";
    unsigned char msg5[] = "                   Result = ";
//
// Configure the USART
//
    Usart_Init(9600);                              // Baud=9600
//
// Program loop
//
    for(;;)                                        // Endless loop
    {
      MyKey = 0;
      Op1 = 0;
      Op2 = 0;

      Newline();                                   // Send newline
      Newline();                                   // Send newline
      Text_To_Usart(msg1);                         // Send TEXT
      Newline();                                   // Send newline
      Newline();                                   // Send newline

//
// Get the first number
//
      Text_To_Usart(msg2);                         // Send TEXT to USART
      do                                           // Get first number
      {
        if(Usart_Data_Ready())                     // If a character ready
        {
          MyKey = Usart_Read();                    // Get a character
          if(MyKey == Enter)break;                 // If ENTER key
          Usart_Write(MyKey);                      // Echo the character
          Key = MyKey - '0';
          Op1 = 10*Op1 + Key;                      // First number in Op1
        }
      }while(1);
      Newline();

//
// Get the second character
//
      Text_To_Usart(msg3);                         // Send TEXT to USART
      do                                           // Get second number
      {
        if(Usart_Data_Ready())
        {
          MyKey = Usart_Read();                    // Get a character
          if(Mykey == Enter)break;                 // If ENTER key
          Usart_Write(MyKey);                      // Echo the character
```

图6-56 （续）

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源


```
            Key = MyKey - '0';
            Op2 = 10*Op2 + Key;                           // Second number in Op2
        }
       }while(1);

      Newline();
//
// Get the operation
//
      Text_To_Usart(msg4);
      do
      {
        if(Usart_Data_Ready())
        {
            MyKey = Usart_Read();                         // Get a character
            if(MyKey == Enter)break;                      // If ENTER key
            Usart_Write(MyKey);                           // Echo the character
            Key = MyKey;
        }
      }while(1);

//
// Perform the operation
//
      Newline();
      switch(Key)                                         // Calculate
      {
        case Plus:
              Calc = Op1 + Op2;                           // If ADD
              break;
        case Minus:
              Calc = Op1 - Op2;                           // If Subtract
              break;
        case Multiply:
              Calc = Op1 * Op2;                           // If Multiply
              break;
        case Divide:
              Calc = Op1 / Op2;                           // If Divide
              break;
      }

      LongToStr(Calc, op);                                // Convert to string
//
// Remove leading blanks
//
      j=0;
      for(i=0;i<=11;i++)
      {
        if(op[i] != ' ')                                  // If a blank
        {
          kbd[j]=op[i];
          j++;
        }
      }

      Text_To_Usart(msg5);
      for(i=0; i<j;i++)Usart_Write(kbd[i]);               // Display result

    }
}
```

图6-56 （续）

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

tyw藏书

在程序的开始处，将程序中用到的不同信息分别定义为msg1~msg5。然后使用mikroC库子程序Usart_Init将USART的波特率初始化为9 600。然后在PC显示器上显示标题“CALCULATOR PROGRAM”。程序使用库函数Usart_Read从键盘读入第一个数。函数Usart_Data_Ready用来检查一个新的数据字节在被读取之前是否已准备好。变量Op1用来存储第一个数。同样地，在另一个循环中，读入第二个数并存储到变量Op2。接下来，程序读入要执行的运算操作（+−×/）。所需的运算操作在一个switch语句中进行，所得的运算结果被保存到变量Calc。程序然后调用库函数LongToStr，将运算结果转换为字符串格式。去除字符串前面的空白，将最终的结果存储到字符数组kbd，并发送到USART以在PC显示器上进行显示。 359

测试程序

程序可使用终端模拟软件来测试，如HyperTerminal，它是Windows操作系统提供的免费工具。具体的程序测试步骤（假设使用串行端口COM2）如下： 360~363

- 将来自微控制器的RS232输出连接至PC的串行输入端口（如COM2）；
- 启动HyperTerminal终端模拟软件，并给本次会话命名；
- 选择 File→New connection→Connect using，选择COM2；
- 选择波特率为9 600、数据位为8、无奇偶位、停止位为1；
- 复位微控制器。

HyperTerminal屏幕的输出示例如图6-57所示。

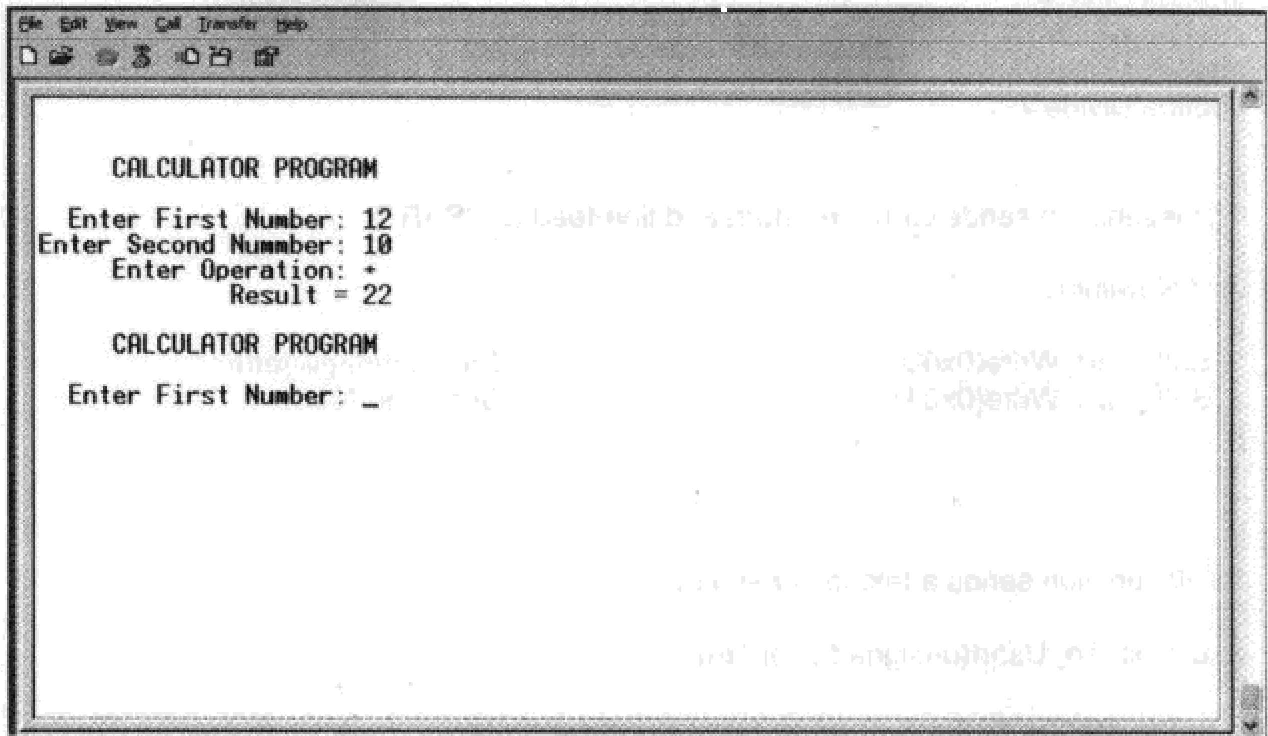


图6-57 HyperTerminal屏幕

使用基于软件的串行通信

前述的例子使用了微控制器的USART和特殊的串行I/O引脚。串行通信也可以完全通过软件方式来实现，而不使用USART。在这种方法里，微控制器的任何引脚都可以用于串行通信。 364

对于项目10给出的计算器程序，可以使用mikroC软件串行通信库函数Software Uart Library来对其进行重新编程。

修改过的程序清单如图6-58所示。项目的电路原理图与图6-54所示的相同（即RC6和RC7分别用作串行TX和RX），虽然也可以使用其他的端口引脚。在程序的开始处，通过调用函数Soft_Uart_Init来配置串行的I/O端口。串行端口名、用作TX和RX的引脚、波特率和通信模式都需要详细地指明。通信模式用来说明微控制器数据是否需要取反。将模式设置为1，表示需要取反数据。当使用MAX232芯片时，数据是不需要取反的（即模式为0）。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
/*****************************************************************************
                    CALCULATOR WITH PC INTERFACE
                    ============================

In this project a PC is connected to a PIC18F452 microcontroller. The project is a
simple integer calculator. User enters the numbers using the PC keyboard. Results are
displayed on the PC monitor.

The following operations can be performed:

  + - * /

In this program the serial communication is handled in software
and the serial port is configured to operate with 9600 Baud rate.

Port pins RC6 and RC7 are used for serial TX and RX respectively.


Author:      Dogan Ibrahim
Date:        July 2007
File:        SERIAL2.C
*****************************************************************************/

#define Enter 13
#define Plus '+'
#define Minus '-'
#define Multiply '*'
#define Divide '/'

//
// This function sends carriage-return and line-feed to USART
//
void Newline()
{
  Soft_Uart_Write(0x0D);                       // Send carriage-return
  Soft_Uart_Write(0x0A);                       // Send line-feed
}


//
// This function sends a text to serial port
//
void Text_To_Usart(unsigned char *m)
{
  unsigned char i;

  i = 0;
  while(m[i] != 0)
  {                                            // Send TEXT to serial port
    Soft_Uart_Write(m[i]);
    i++;
  }
}


//
// Start of MAIN program
//
void main()
```

图6-58 修改过的程序

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
{
    unsigned char MyKey, i,j,error,kbd[5],op[12];
    unsigned long Calc, Op1, Op2,Key;
    unsigned char msg1[] = "    CALCULATOR PROGRAM";
    unsigned char msg2[] = "  Enter First Number: ";
    unsigned char msg3[]= "Enter Second Nummber: ";
    unsigned char msg4[] = "    Enter Operation: ";
    unsigned char msg5[] = "            Result = ";
//
// Configure the serial port
//
    Soft_Uart_Init(PORTC,7,6,2400,0);          // TX=RC6, RX=RC7, Baud=9600
//
// Program loop
//
    for(;;)                                    // Endless loop
    {
      MyKey = 0;
      Op1 = 0;
      Op2 = 0;

      Newline();                               // Send newline
      Newline();                               // Send newline
      Text_To_Usart(msg1);                     // Send TEXT
      Newline();                               // Send newline
      Newline();                               // Send newline

//
// Get the first number
//
      Text_To_Usart(msg2);                            // Send TEXT
      do                                              // Get first number
      {
        do                                            // If a character ready
          MyKey = Soft_Uart_Read(&error);             // Get a character
        while (error);
        if(MyKey == Enter)break;                      // If ENTER key
        Soft_Uart_Write(MyKey);                       // Echo the character
        Key = MyKey - '0';
        Op1 = 10*Op1 + Key;                           // First number in Op1
      }while(1);

      Newline();

  //
  // Get the second character
  //
        Text_To_Usart(msg3);                          // Send TEXT
        do                                            // Get second number
        {
          do
            MyKey = Soft_Uart_Read(&error);           // Get a character
          while(error);
          if(Mykey == Enter)break;                    // If ENTER key
          Soft_Uart_Write(MyKey);                     // Echo the character
          Key = MyKey - '0';
          Op2 = 10*Op2 + Key;                         // Second number in Op2

        }while(1);
```

图6-58 （续）

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
            Newline();
    //
    // Get the operation
    //
            Text_To_Usart(msg4);
            do
            {
              do
                 MyKey = Soft_Uart_Read(&error);          // Get a character
              while(error);
              if(MyKey == Enter)break;                    // If ENTER key
              Soft_Uart_Write(MyKey);                     // Echo the character
              Key = MyKey;

            }while(1);

    //
    // Perform the operation
    //
           Newline();
           switch(Key)                                    // Calculate
           {
            case Plus:
                  Calc = Op1 + Op2;                       // If ADD
                  break;
            case Minus:
                  Calc = Op1 - Op2;                       // If Subtract
                  break;
            case Multiply:
                  Calc = Op1 * Op2;                       // If Multiply
                  break;
            case Divide:
                  Calc = Op1 / Op2;                       // If Divide
                  break;
           }

           LongToStr(Calc, op);                           // Convert to string
    //
    // Remove leading blanks
    //
           j=0;
           for(i=0;i<=11;i++)
           {
             if(op[i] != ' ')                              // If a blank
             {
               kbd[j]=op[i];
               j++;
             }
           }

           Text_To_Usart(msg5);
           for(i=0; i<j;i++)Soft_Uart_Write(kbd[i]);      // Display result

         }
    }
```

图6-58 （续）

365
~
369

串行数据的输出使用函数Soft_Uart_Write，串行数据的输入使用函数Soft_Uart_Read。由于读取操作是一个无阻塞函数，所以在执行读取操作之前，很有必要检查数据字节是否可用。使用函数的error变量可以完成这个工作。程序的其余部分同项目10。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com
tyw藏书

第 7 章　高级 PIC18 项目——SD 卡项目

从本章起，将陆续介绍基于PIC18微控制器的更为复杂的项目设计。本章将讨论基于SD（Secure Digital，安全数字）存储卡的项目设计。本书后面几章还会陆续讨论基于流行USB总线和CAN总线协议的项目的基本理论和设计。

7.1　SD 卡

在介绍基于SD卡项目的设计细节之前，首先来学习SD卡存储设备的基本原理和操作方法。图7-1给出了一种常见的SD卡。



图7-1　常见的SD卡

SD卡是一种闪存设备，能够在小体积上实现大容量的、非易失性的、可重写的存储。这些设备广泛地应用于许多电子消费品中，如数码相机、电脑、GPS系统、移动电话、PDA等。SD卡的存储容量也在与日俱增。目前，SD卡的容量已经从256 MB增加到了8 GB。SD卡有三种规格：标准型、迷你型和微小型。表7-1列出了最通用的标准型SD卡和迷你型SD卡的主要规范。

SD卡的规范是由SD卡协会（会员人数超过600个）制定的。迷你型SD卡和微小型SD卡在电气特性上与标准型SD卡是兼容的，可以插入到专门的适配器中使用。插在标准型的卡插槽中，可以用作标准型SD卡。

PDG

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

表7-1 标准型SD卡和迷你型SD卡

	标准型SD卡	迷你型SD卡
尺寸	32×24×2.1 mm	21.5×20×1.4 mm
卡重	2.0g	1.0g
工作电压	2.7 V～36 V	2.7 V～36 V
写保护	是	否
引脚数目	9	11
接口类型	SD或SPI	SD或SPI
电流消耗	<75 mA(写)	<40 mA(写)

372

SD卡的速度可用三种方式来度量：KB/s（千比特每秒）、MB/s（兆比特每秒），还有就是使用类似于度量CD-ROMS速度的“x”额定值，其中“x”相当于150KB/s的速度。不同的基于“x”度量的速度有：

- 4x：600 KB/s
- 16x：2.4 MB/s
- 40x：6.0 MB/s
- 66x：10 MB/s

本章只使用标准型SD卡。更小的SD卡的规范和标准SD卡的规范是相同的，本章将不做更深入的描述。

SD卡与微控制器的接口可以使用两种不同的协议：SD卡协议和SPI（Serial Peripheral Interface，串行外设接口）协议。本章只使用更流行的SPI协议。标准SD卡有9个引脚，引脚布局图如图7-2所示。根据接口协议，这些引脚有着不同的功能。表7-2给出了在SD和SPI操作模式下每个引脚的功能。

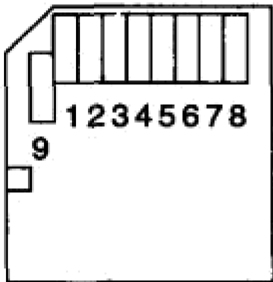


图7-2 标准型SD卡的引脚布局

表7-2 标准型SD卡的引脚定义

引脚编号	引脚名称	SD描述	SPI描述
1	CD/DAT3/CS	数据线3	片选
2	CMD/Datain	命令/应答	卡命令和数据主机
3	VSS	电源地	电源地
4	VDD	电源电压	电源电压
5	CLK	时钟	时钟
6	VSS2	电源电压参考地	电源电压参考地
7	DAT0	数据线0	控制数据和状态的卡
8	DAT1	数据线1	保留
9	DAT2	数据线2	保留

PDG

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

因为本章描述的项目是基于SPI总线协议的，所以在进行项目设计之前，不妨先了解一下SPI总线规范。

7.1.1 SPI 总线

SPI总线是一种同步的串行总线标准，由Motorola命名，可工作于全双工模式。使用SPI总线的设备可在主从模式下工作，其中，主设备初始化数据传输、选择从设备，并为从设备提供时钟。被选择的从设备进行应答，并在每个时钟脉冲下向主设备发送数据。SPI总线可以与一个主设备和一个或多个从设备一起工作。这种简单的接口，又被称为“四线”接口。 373

SPI总线上的信号命名如下：

- MOSI——主设备输出，从设备输入
- MISO——主设备输入，从设备输出
- SCLK——串行时钟
- SS——从设备选择

这些信号也可以被命名为：

- DO——数据输出
- DI——数据输入
- CLK——时钟
- CD——片选

图7-3给出了在SPI总线上主从设备之间的基本连接。主设备通过数据线MOSI发送数据，通过数据线MISO接收数据。在数据传输之前必须先选择从设备。 374

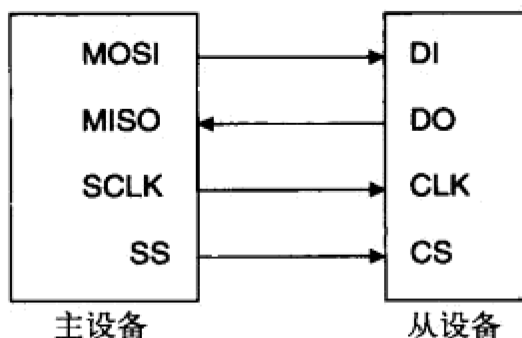


图7-3 SPI主从方式连接

图7-4给出了一个将多个从设备连接到SPI总线的例子。在这里，主设备可以单独地选择每一个从设备，尽管所有的从设备都能接收时钟脉冲，但是只有被选中的从设备才会作出应答。如果一个SPI设备没有被选中，那么它的数据输出是高阻态的，因此它不会在总线上妨碍当前被选中的从设备。 375

当主设备提供时钟脉冲时，数据可以正常地在主从设备之间进行输入或者输出。在开始通信之前，主设备首先需要将从设备的片选线置为低电平。然后主设备发出时钟脉冲，并在每一个时钟周期内，进行一次全双工的数据传输。当数据全部传输完毕后，主设备停止时钟输出。

目前，在使用SPI总线的微控制器接口电路中，可以连接一系列的设备，例如：

- 存储设备（SD卡）
- 传感器
- 实时时钟
- 通信设备
- 显示器

PDG

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

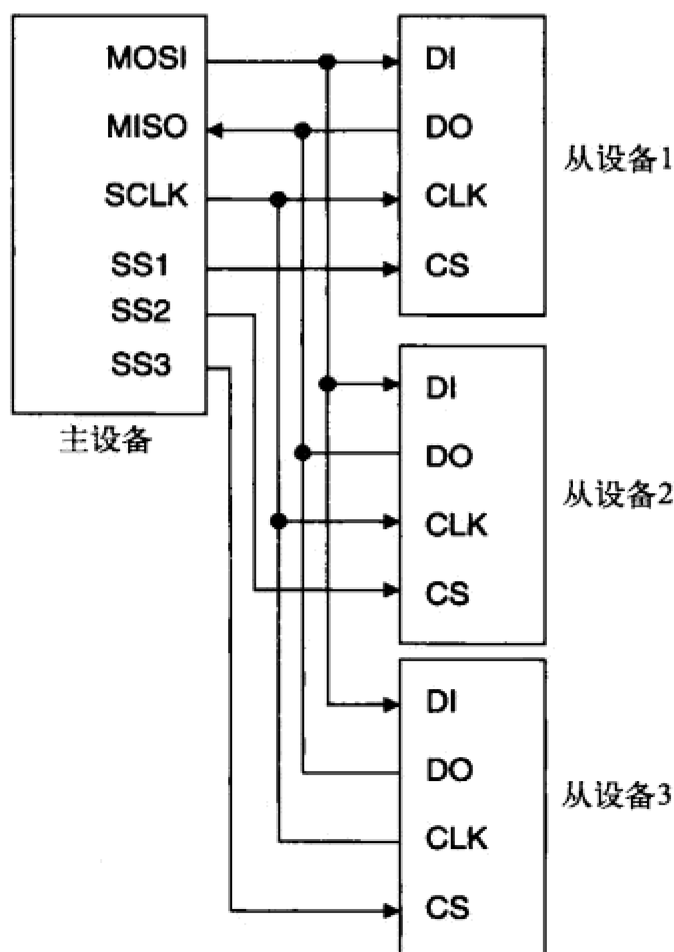


图7-4　多个从设备的SPI总线

SPI总线具有如下的优点：

- 简单的通信协议
- 全双工通信
- 非常简单的硬件接口

而SPI总线也有缺点：

- 需要4个引脚
- 没有硬件溢出控制
- 没有从设备确认

值得注意的是，国际性委员会还没有任何的SPI标准，因此，SPI总线实现存在好几个版本。在一些应用中，将MOSI和MISO线组合成一条单独的数据线，从而将线缆的需求量减少到了3根。有些实现提供两个时钟，一个时钟用来捕获（或者显示）数据，另一个时钟用来向设备提供时钟信号。同样地，在有些实现中，片选线可以是高电平有效，而不是低电平有效。

376

7.1.2　在SPI模式下SD卡的操作

当SD卡工作在SPI模式下时，只有7个引脚被使用。3个引脚（即引脚3、引脚4和引脚6）用于提供电源，而其余的4个引脚（即引脚1、引脚2、引脚5和引脚7）则用于SPI模式下的操作：

- 两个电源地（引脚3和引脚6）
- 工作电源（引脚4）
- 片选（引脚1）
- 数据输出（引脚7）
- 数据输入（引脚2）
- 时钟信号（引脚5）

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

上电后，SD卡默认为SD总线协议。如果在接收复位指令期间发出片选（CS）信号，那么SD卡将切换到SPI模式。当SD卡工作在SPI模式时，它只对SPI指令作出应答。主机可以通过将电源先关闭再打开来复位SD卡。

mikroC编译器提供了一个指令库，用来对SD卡进行初始化和读写操作。因为可以使用库里的函数，所以在使用SD卡之前，不必详细地知道SD卡的内部结构。然而，对SD卡的内部结构有一个基本的了解，可以更好地帮助读者使用SD卡。本节将简要地介绍SD卡的内部结构和操作方法。

SD卡带有一组寄存器，用来提供该SD卡的状态信息。当SD卡工作在SPI模式时，这组寄存器是：

- 卡识别寄存器（CID）
- 卡特定数据寄存器（CSD）
- SD配置寄存器（SCR）
- 操作控制寄存器（OCR）

CID寄存器由16 B组成，包含制造商ID、产品名称、产品修订、卡序列号、生产日期编码，以及校验和字节等信息。表7-3给出了CID寄存器的结构。

377

表7-3 CID寄存器的结构

名 称	数据类型	字 长	注 释
制造商ID（MID）	二进制码	1B	制造商ID（如SanDisk为0x03）
OEM/应用ID（ODID）	ASCII码	2B	识别卡的OEM和/或卡内容
产品名称（PNM）	ASCII码	5B	产品名称
产品修订（PRV）	BCD码	1B	两个二进制编码的数字
序列号（PSN）	二进制码	4B	32位无符号整数
保留		4B	高4位
厂商日期编码（MDT）	BCD码	12B	生产日期（与2000年的差值）
CRC-7校验和	二进制码	7B	校验和
未使用	二进制码	1B	总是1

CSD寄存器由16 B组成，包含诸如卡数据传输速率、读/写块长度、读/写电流、可擦除块大小、文件格式、写保护标志和校验和（checksum）等卡的详细信息。表7-4给出了CSD寄存器的结构。

表7-4 CSD寄存器的结构

字 节	
字节0	0 0 ××××××
字节1	TAAC[7:0]
字节2	NSAC[7:0]
字节3	TRAN_SPEED[7:0]
字节4	CCC[11:4]
字节5	CCC[3:0]READ_BL_LEN[3:0]
字节6	READ_BL_PARTIAL WRITE_BLK_MISALIGN READ_BLK_MISALIGN DSR_IMP××C_SIZE(11:10)
字节7	C_SIZE[9:2]
字节8	C_SIZE[1:0]VDD_R_CURR_MIN(2:0)VDD_R_CURR_MAX(2:0)
字节9	VDD_W_CURR_MIN(2:0)VDD_W_CURR_MAX(2:0) C_SIZE_MULT(2:1)
字节10	ERASE_BLK_EN SECTOR_SIZE(6:1)

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

（续）

字　节	
字节11	SECTOR_SIZE(0)WP_GRP_SIZE(6:0)
字节12	WP_GRP_ENABLE××R2W_FACTOR(2:0)
字节13	WRITE_BL_LEN(1:0)0×××××
字节14	FILE_FORMAT_GRP COPY PERM_WRITE_PROTECT TMP_WRITE_ PROTECT FILE_FORMAT(1:0)××
字节15	CRC(6:0)1

字段定义	
TAAC	读取数据访问时间1（如1.5ms）
NSAC	在CLK周期读取数据访问时间
TRAN_SPEED	最大数据传输速率
CCC	卡指令类型
READ_BL_LEN	最大的读取数据块长度（如512B）
READ_BL_PARTIAL	允许读取的部分块
WRITE_BLK_MISALIGN	写块不对齐
READ_BLK_MISALIGN	读块不对齐
DSR_IMP	DSR实现
C_SIZE	设备尺寸
VDD_R_CURR_MIN	在VDD最小时的最大读取电流
VDD_R_CURR_MAX	在VDD最大时的最大读取电流
VDD_W_CURR_MIN	在VDD最小时的最大写入电流
VDD_W_CURR_MAX	在VDD最大时的最大写入电流
C_SIZE_MULT	设备尺寸系数
ERASE_BLK_EN	允许擦除单个块
SECTOR_SIZE	擦除扇区大小
WP_GRP_SIZE	写保护组大小
WP_GRP_ENABLE	允许写保护组
R2W_FACTOR	写速度系数
WRITE_BL_LEN	最大的写数据块长度（如512B）
WRITE_BL_PARTIAL	允许写入的部分块
FILE_FORMAT_GRP	文件格式组
COPY	复制标志
PERM_WRITE_PROTECT	永久性写入保护
TMP_WRITE_PROTECT	临时性写入保护
FILE_FORMAT	文件格式

SCR寄存器有8 B长，包含SD卡的特殊功能和特性，如安全支持功能和支持的数据总线宽度等。

OCR寄存器只有4 B长，用来存储SD卡的VDD电压范围。OCR寄存器将给出一个电压范围，在此范围内卡才可以对数据进行存取。

所有SD卡的SPI指令都是6 B长的，并且MSB先传输。图7-5给出了指令的格式。第一个字节是指令字节，其余的5个字节叫作指令参数。指令字节的第六位置为1，而MSB位则总是0。使用剩下的6位，可以得到64个可能的指令，分别被命名为CMD0～CMD63。下面是部分重要的

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

指令：

- CMD0 GO_IDLE_STATE（复位SD卡）
- CMD1 SEND_OP_COND（初始化SD卡）
- CMD9 SEND_CSD（获取CSD寄存器数据）
- CMD10 SEND_CID（获取CID寄存器数据）
- CMD16 SET_BLOCKLEN（选择块大小，以字节为单位）
- CMD17 READ_SINGLE_BLOCK（读取数据块）
- CMD24 WRITE_BLOCK（写入块数据）
- CMD32 ERASE_WR_BLK_START_ADDR（设置第一个要擦除的写入块的地址）
- CMD33 ERASE_WR_BLK_END_ADDR（设置最后一个要擦除的写入块的地址）
- CMD38 ERASE（擦除所有已选择的块）

378～380

字节1			字节2～5	字节6	
7	6		31　　0	7	0
0	1	指令	指令参数	CRC	1

图7-5　SD卡的SPI指令格式

在应答一个指令时，SD卡将会发送一个状态字节R1。该字节的MSB位总是0，而其他位则用来表明如下的错误状态：

- 卡处于空闲状态
- 擦除复位
- 非法指令
- 通信CRC错误
- 擦除序列错误
- 地址错误
- 参数错误

1. 读数据

在SPI模式下，SD卡提供单块和多块的读操作。主设备应该设置块的长度。在执行一个有效的读指令之后，SD卡将使用一个应答标记来作出应答，接下来是一个数据块和CRC检测。块的长度可以在1B～512B之间不等。起始地址可以是卡地址范围内的任何有效地址。

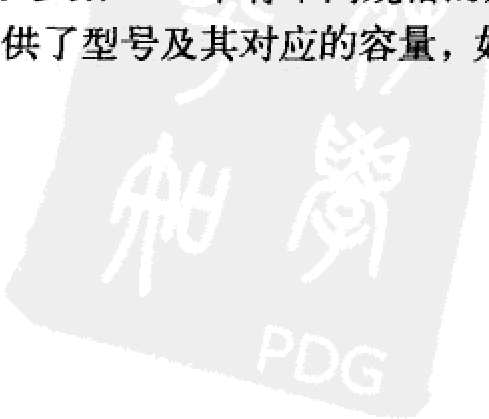
在多块的读操作中，SD卡发送数据块，每一个数据块在块的末尾都有它们自己的CRC检测。 381

2. 写数据

在SPI模式下，SD卡可提供单块或多块的写操作。在从主设备接收到一条有效的写指令后，SD卡使用一个应答标记来作出应答，然后等待接收数据块。在每个数据块的起始处添加有一个块开始标记，长度为1B。在接收到数据块后，SD卡使用一个数据应答标记来作出应答，只要接收到的数据块没有错误，SD卡就被编程成功。

在多块写操作中，主设备一个数据块接一个数据块地发送数据，每个数据块之前都添加有一个块开始标记。在接收完每一个数据块后，SD卡会发送一个应答字节。

卡的大小参数　SD卡有不同规格的尺寸。在执行写操作的时候，SanDisk公司（www.sandisk.com）提供了型号及其对应的容量，如表7-5所示。SanDisk公司现在可以提供4 GB甚至更大容量的型号。



世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

tyw藏书

表7-5 SanDisk卡的型号和容量

型 号	容 量
SDSDB-16	16 MB
SDSDB-32	32 MB
SDSDJ-64	64 MB
SDSDJ-128	128 MB
SDSDJ-256	256 MB
SDSDJ-512	512 MB
SDSDJ-1024	1 024 MB

在SD卡上除了正常的存储区域以外，还有一个关于安全版权管理的受保护区域。这个区域可供应用来保存安全相关的数据，主设备可使用安全读/写指令来访问。卡的写保护机制对这个区域没有影响。表7-6给出了受保护区域的大小和可供用户读/写数据的数据区域大小。例如，对于一个1GB的卡，保护区域是20 480个块（每一个块是512 B），而用户数据区域是1 983 744个块。

382

表7-6 保护区域和数据区域大小

型 号	保护区域（块）	用户区域（块）
SDSDB-16	352	28 800
SDSDB-32	736	59 776
SDSDJ-64	1 376	121 856
SDSDJ-128	2 624	246 016
SDSDJ-256	5 376	494 080
SDSDJ-512	10 240	940 864
SDSDJ-1024	20 480	1 983 744

1块=512B

使用原始扇区访问方法，可以对卡的任意扇区进行读取或者写入数据。通常，SD卡数据都被构造为一个文件系统，在卡上设置两个DOS格式的分区：用户区域和安全保护区域。表7-7给出了每个区域的大小。例如，在1 GB的卡中，安全保护区域的大小为519个扇区（每个扇区是512 B），而用户数据区域的大小为1 982 976个扇区。

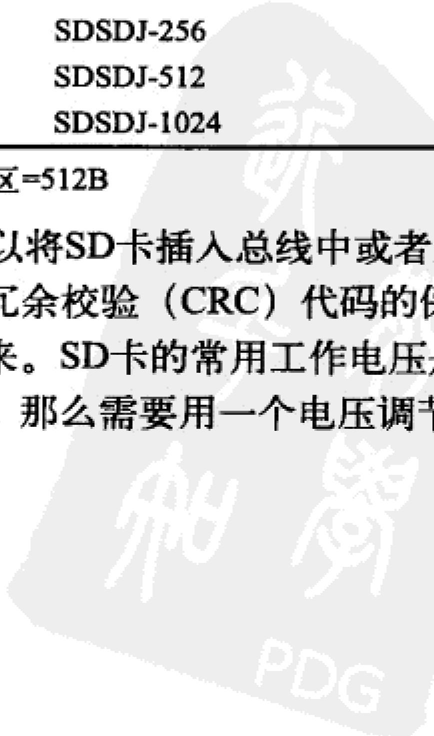
表7-7 在DOS格式的卡中安全保护区域和用户数据区域的大小

型 号	保护区域（扇区）	用户区域（扇区）
SDSDB-16	39	28 704
SDSDB-32	45	59 680
SDSDJ-64	57	121 760
SDSDJ-128	95	245 824
SDSDJ-256	155	493 824
SDSDJ-512	275	990 352
SDSDJ-1024	519	1 982 976

383

1扇区=512B

可以将SD卡插入总线中或者从总线中移除，而不会损坏。这是因为所有的数据传输操作都受循环冗余校验（CRC）代码的保护，并且因插入或移除卡而导致任何的位改变都会被轻易地检测出来。SD卡的常用工作电压是2.7 V。允许的最大工作电压为3.6 V。如果使用标准的5 V电压供电，那么需要用一个电压调节器来将电压降低到2.7 V。



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

使用SD卡，需要将卡插入到专门的带有外部接头的卡座（如图7-6所示），这样就很容易连接到需要的卡引脚了。

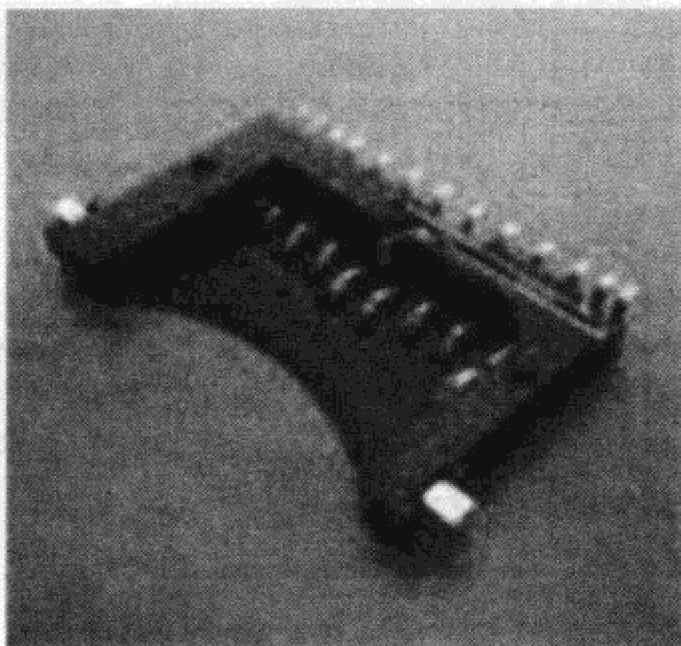


图7-6 SD卡插座

7.2 mikroC 语言的 SD 卡库函数

mikroC语言提供了一个庞大的库函数集来对SD（也可以是MultiMediaCards，MMC）卡进行读写数据。可以对卡的给定扇区进行数据读写操作，卡上的文件系统也可以在更复杂的应用中使用。

下面是mikroC提供的库函数：

384

- Mmc_Init （初始化卡）
- Mmc_Read_Sector （读一个扇区的数据）
- Mmc_Write_Sector （写一个扇区的数据）
- Mmc_Read_Cid （读CID寄存器）
- Mmc_Read_Csd （读CSD寄存器）
- Mmc_Fat_Init （初始化FAT）
- Mmc_Fat_QuickFormat （将卡格式化为FAT16）
- Mmc_Fat_Assign （分配正在使用的文件）
- Mmc_Fat_Reset （复位文件指针，打开当前分配的文件用来读）
- Mmc_Fat_Rewrite （复位文件指针，并清除分配的文件；打开分配的文件用来写）
- Mmc_Fat_Append （将文件指针移动到分配的文件末尾，准备向文件添加新的数据）
- Mmc_Fat_Read （读文件指针指向的字节）
- Mmc_Fat_Write （向分配的文件中写入一个块数据）
- Mmc_Set_File_Date （向文件中写入系统时间）
- Mmc_Fat_Delete （删除文件）
- Mmc_Fat_Get_File_Date （读文件的系统时间）
- Mmc_Fat_Get_File_Size （获取文件大小，单位为字节）
- Mmc_Fat_Get_Swap_File （创建一个交换文件）

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

在本章的剩余部分，将介绍一些SD卡和基于PIC18微控制器的项目。

项目7.1　读CID寄存器并在PC屏幕上显示

385

在这个项目中，SD卡连接到PIC18F452型微控制器上。微控制器的串行输出端口连接到PC的串行输入端口（如COM1）。微控制器读取卡的CID寄存器中的内容，并将数据发送给PC，这样数据就可以显示在PC的屏幕上了。

图7-7给出了项目的方框图。

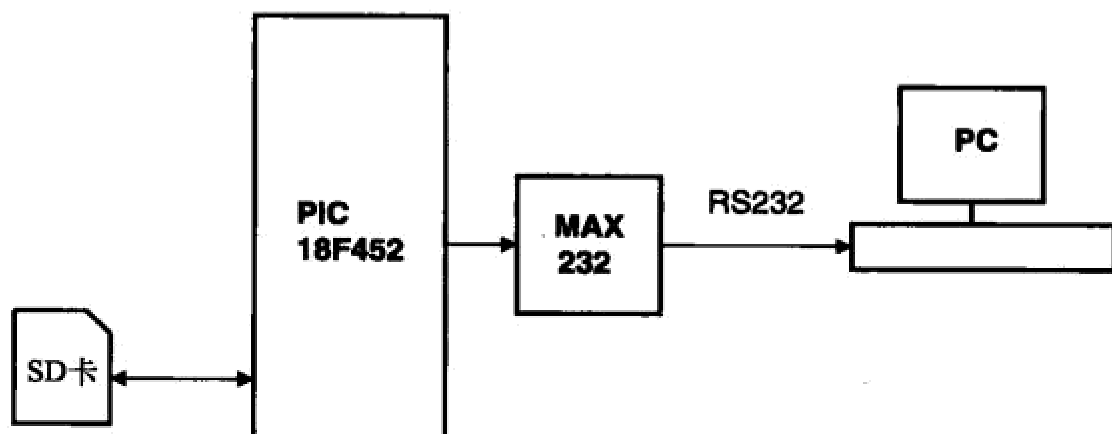


图7-7　项目的方框图

该项目的电路原理图如图7-8所示。将SD卡插入一个卡座，然后通过2.2kΩ和3.3kΩ的电阻连接到PIC18F452微控制器的PORTC端口，使用到的引脚如下：

- 卡的CS，连接到PORTC的RC2引脚
- 卡的CLK，连接到PORTC的RC3引脚
- 卡的DO，连接到PORTC的RC4引脚
- 卡的DI，连接到PORTC的RC5引脚

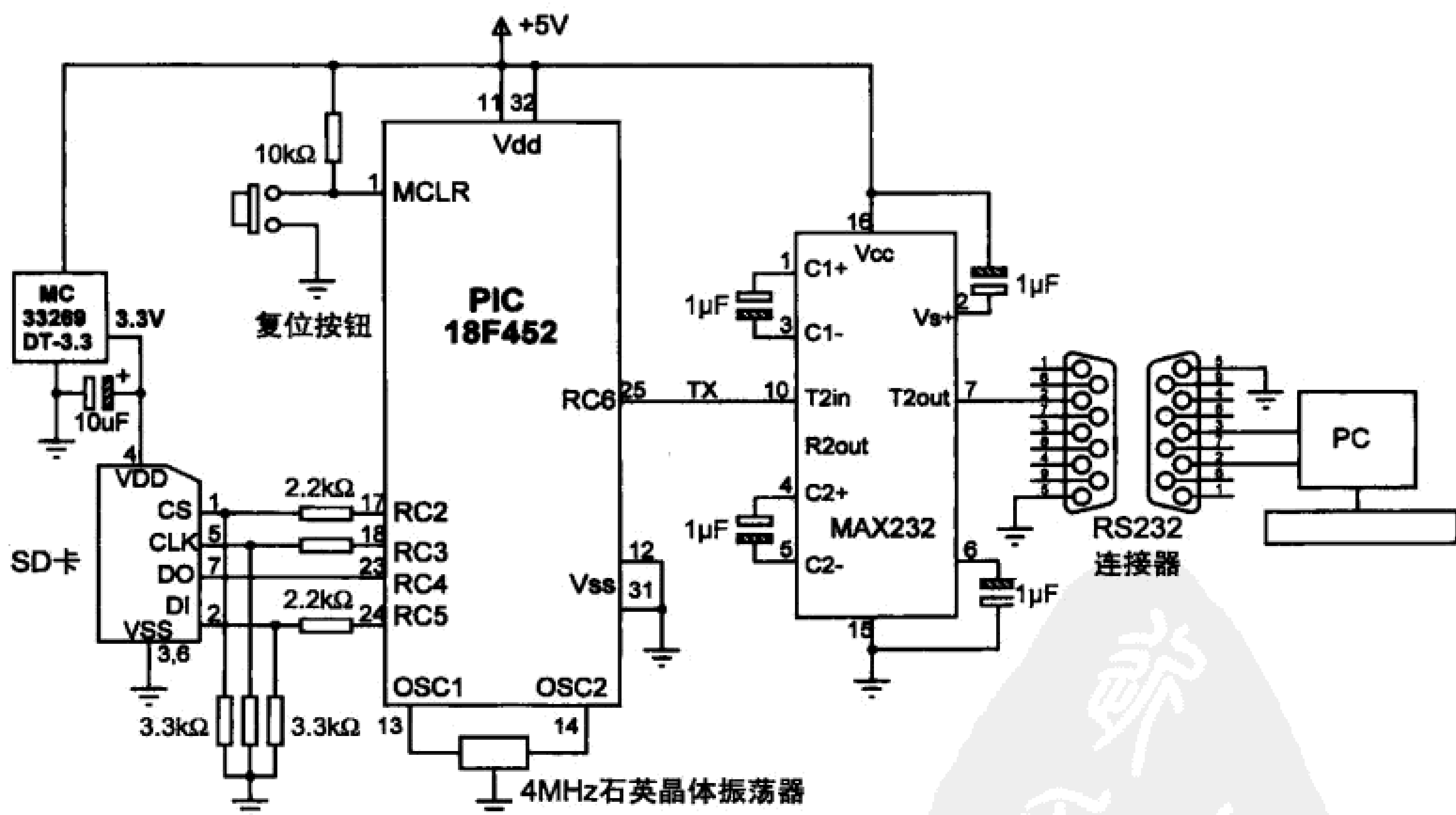


图7-8　项目的电路原理图

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

tyw藏书

根据SD卡的规范，当卡工作在VDD=3.3 V电压时，输入输出引脚的电压水平如下：

- 输出高电平的最小电压，VOH=2.475 V；
- 输出低电平的最大电压，VOL=0.4125 V；
- 输入高电平的最小电压，VIH=2.0625 V；
- 输入高电平的最大电压，VIH=3.6 V；
- 输入低电平的最大电压，VIL=0.825 V。

386

虽然由卡产生的输出（2.475 V）足以驱动PIC微控制器的输入端口，但是微处理器的逻辑高电平输出（大约4.3 V）对于SD卡的输入（最高3.6 V）来说还是太高了。因此，在SD卡的3个输入端，使用了2.2kΩ和3.3kΩ的电阻来进行分压。这样，就可以将SD卡输入端的最大电压限制在2.5V左右：

$$\text{SD卡输入电压}=4.3\ \text{V}\times 3.3\ \text{k}\Omega/(2.2\ \text{k}\Omega+3.3\ \text{k}\Omega)=2.48\ \text{V}$$

微控制器的串行输出端口引脚RC6（TX）连接到一个MAX232类型的RS232电压转换芯片，然后连接到一个9引脚的D型连接器。这样就可以连接到PC的串行输入端口了。

微控制器由5 V电源供电，这可以通过一个7805类型的5 V调节器从9 V的输入端获得。SD卡要求的供电电压2.7 V～3.6 V，可以通过MC33269DT-3.3调节器来得到，该调节器的输入电压是5 V，输出是3.3 V。

项目的程序清单如图7-9（程序SD1.C）所示。在主程序的开始，将字符数组CID声明为16个字节长度。

387

```
/*******************************************************************
                    SD CARD PROJECT
                    ===============

In this project a SD card is connected to PORTC as follows:

  CS    RC2
  CLK   RC3
  DO    RC4
  DI    RC5

In addition, a MAX232 type RS232 voltage level converter chip
is connected to serial output port RC6.

The program reads the SD card CID register parameters and
sends it to a PC via the serial interface. This process is
repeated at every 10 seconds.

The UART is set to operate at 2400 Baud, 8 bits, no parity.

Author:   Dogan Ibrahim
Date:     August 2007
File:     SD1.C
*******************************************************************/

//
// This function sends carriage-return and line-feed to USART
//
void Newline()
{
   Soft_Uart_Write(0x0D);              // Send carriage-return
   Soft_Uart_Write(0x0A);              // Send line-feed
```

图7-9 程序清单

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
}


//
// This function sends a space character to USART
//
void Space()
{
   Soft_Uart_Write(0x20);
}


//
// This function sends a text to serial port
//
void Text_To_Usart(unsigned char *m)
{
   unsigned char i;
   i = 0;
   while(m[i] != 0)
   {                                          // Send TEXT to serial port
     Soft_Uart_Write(m[i]);
     i++;
   }
}

//
// This function sends string to serial port. The string length is passed as an argument
//
void Str_To_Usart(unsigned char *m,unsigned char 1)
{
    unsigned char i;
    unsigned char txt[4];

    i=0;
    for(i=0; i<l; i++)
    {
      ByteToStr(m[i],txt);
      Text_To_Usart(txt);
      Space();
    }
}


//
// Start of MAIN program
//
void main()
{
    unsigned char error,CID[16];
    unsigned char msg[] = "    SD CARD CID REGISTER";

//
// Configure the serial port
//
  Soft_Uart_Init(PORTC,7,6,2400,0);                    // TX=RC6
//
// Initialise the SD card
//
   Spi_Init_Advanced(MASTER_OSC_DIV16,DATA_SAMPLE_MIDDLE,
                CLK_IDLE_LOW, LOW_2_HIGH);
```

图7-9 （续）

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
//
// Initialise the SD bus
//
   while(Mmc_Init(&PORTC,2));
//
// Start of MAIN loop. Read the SD card CID register and send the data
// to serial port every 10 seconds
//
   for(;;)                                    // Endless loop
   {
     Text_To_Usart(msg);                      // Send TEXT
     Newline();                                        // Send newline
     Newline();                                        // Send newline
     error = Mmc_Read_Cid(CID);                        // Read CID register into CID
//
// Send the data to RS232 port
//
     Str_To_Usart(CID,16);                    // Send CID contents to UART
     Delay_Ms(10000);                         // Wait 10 seconds
     Newline();
     Newline();
   }
}
```

图7-9 （续）

在系统上电时，将需要在屏幕上显示的信息加载给变量msg。然后，对PORTC的UART进行初始化，波特率为2 400。

在使用SD卡库函数之前，必须调用函数Spi_Init_Advanced，并使用给定的参数。然后调用函数Mmc_Init对SD卡总线进行初始化，这里指定将卡连接到PORTC。程序然后进入一个无穷的循环，每10秒钟重复一次。在循环中，首先显示标题信息，然后显示两个新行字符。接着，程序通过调用函数Mmc_Read_Cid来读取CID寄存器中的内容，然后将数据存储在字符数组CID中。其次，调用函数Str_To_Usart，将该数据发送到串行端口。在循环的结尾，显示两个新行字符，程序等待10秒，然后重复循环。

将设备连接到PC，并在PC上运行HyperTermiual终端模拟程序来测试项目的运行。将通信参数设置为2 400波特率、8个数据位、1个停止位、无奇偶位。图7-10给出了在屏幕上的输出例子。 388～390

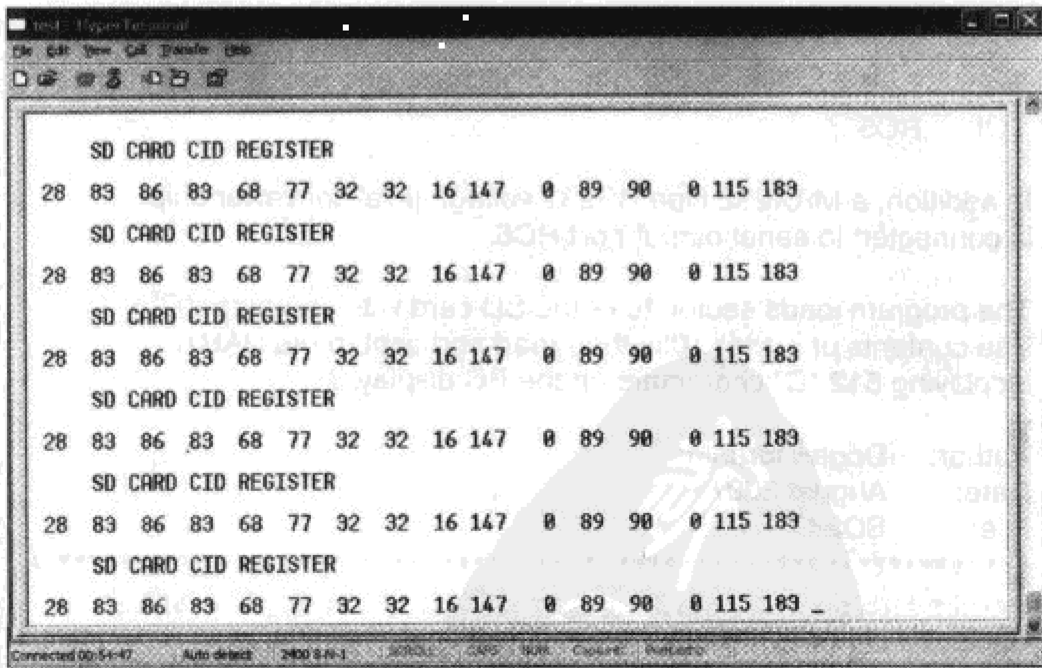


图7-10 在HyperTerminal中项目的输出例子

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源

SD卡返回的数据是：

28 83 86 83 68 77 32 32 16 147 0 89 90 0 115 183

根据表7-3，可以得到卡的如下数据：

制造商ID =28（十进制）
OME/应用ID =SV
产品名称 =SDM
产品修订 =1.0（十进制数16对应于二进制数0001 0000，在BCD码中表示为10，修订号格式为n.m，这里给出的是1.0）
391
序列号 =16 147 0 89（十进制）
保留 =0000位（只有4位）
生产日期代码 =073（这个12位的参数的二进制值为0000 0111 0011，其中最高的4位来自保留字段的低四位，而低8位是十进制数115。这里给出的BCD值是073。日期表示成从2000开始的YYM格式。这样，该卡生产于2007年3月）。
CRC= 1011100（二进制）（LSB位总是1）

项目 7.2 SD 卡扇区的读/写

本项目的硬件和项目7.1（如图7-8所示）是一样的。在本项目中，将SD卡的扇区10填满字符“C”，然后读取这个扇区，并将卡数据发送到UART。

该项目的程序清单如图7-11（程序SD2.C）所示。在程序的开始，声明了两个512B的字符数组data1和data2。使用字符C填充数组data1，然后将该数组的内容写入到SD卡的扇区10。然后，将扇区10的内容读入到字符数组data2，并发送到UART，结果在PC的屏幕上显示512个字符C。在正常情况下，只使用一个数组来读写SD卡。这里使用两个数组是为了说明发送到UART的是卡上的内容，而不是数组data1的内容。

```
/*********************************************************************
                        SD CARD PROJECT
                        ===============

In this project a SD card is connected to PORTC as follows:

  CS    RC2
  CLK   RC3
  DO    RC4
  DI    RC5

In addition, a MAX232 type RS232 voltage level converter chip
is connected to serial output port RC6.

The program loads sector 10 of the SD card with character "C".
The contents of sector 10 is then read and sent to the UART,
displaying 512 "C" characters on the PC display.

Author:   Dogan Ibrahim
Date:     August 2007
File:     SD2.C
*********************************************************************/

unsigned char data1[512],data2[512];
```

图7-11 项目的程序清单

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

kyw藏书

```
unsigned int i;
unsigned short x;

void main()
{

//
// Configure the serial port
//
        Usart_Init(2400);
//
// Initialise the SD card
//  Spi_Init_Advanced(MASTER_OSC_DIV16,DATA_SAMPLE_MIDDLE
                    CLK_IDLE_LOW, LOW_2_HIGH);
//
// Initialise the SD bus
//
   while(Mmc_Init(&PORTC,2));
//
// Fill buffer with character "C"
//
   for(i=0; i<512; i++)data1[i] = 'C';
//
// Write to sector 10
//
   x = Mmc_Write_Sector(10, data1);
//
// Now read from sector 10 into data2 and send to UART
//
   x = Mmc_Read_Sector(10,data2);

   for(i=0; i<400; i++)Usart_Write(data2[i]);        // Send to UART

   for(;;);                                          // Wait here forever
}
```

图7-11 （续）

项目7.3 使用卡文件系统

本项目的硬件和项目7.1（图7-8所示）是一样的。在本项目中，在SD卡上创建一个名为MYFILE55.TXT的文件。首先，将字符串“This is MYFILE.TXT”写入文件。然后，将字符串“This is the added data...”添加到文件。然后，程序读取文件的内容，并且发送字符串“This is MYFILE.TXT. This is the added data...”到UART，使得HyperTerminal运行时能够将数据显示在PC屏幕上。 392

本项目的程序清单如图7-12（程序SD3.C）所示。在程序的开始，将UART的波特率初始化为2 400。然后，根据需要使用库函数对SPI总线和FAT文件系统进行初始化。接下来，程序调用库函数Mmc_Fat_Assign来创建文件MYFILE55.TXT，该库函数使用filename和创建标记0x80作为参变量。如果文件不存在，这些参变量就指示函数创建一个新的文件。尽管可以声明8位数字的文件名和3位数字的扩展名且中间没有“.”（因为“.”将由函数插入），但是文件名的格式还是应该为“文件名.扩展名”。其他创建标记的允许值如表7-8所示。注意到，在读写SD卡之前，必须将其格式化为FAT16。虽然大多数的新卡都已经被格式化，但是也可以调用函数Mmc_Fat_QuickFormat来格式化SD卡。

调用函数Mmc_Fat_Rewrite，将文件清空（如果文件是非空的），然后调用函数Mmc_Fat_Write将字符串“This is MYFILE.TXT”写入文件。注意到，所写入数据的大小必须指定为该函数的第二个参变量。接下来调用函数Mmc_Fat_Append，将第二个字符串“This is the added data...”添加到文件。调用 393

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

394 函数Mmc_Fat_Reset，将文件指针指向文件的开头，并且返回文件的大小。最后，使用一个for循环，调用函数Mmc_Fat_Read，从文件中读取每一个字符，并且调用函数Usart_Write将读取的字符发送到UART。

```
/*****************************************************************
                         SD CARD PROJECT
                         ===============

In this project a SD card is connected to PORTC as follows:

  CS    RC2
  CLK   RC3
  DO    RC4
  DI    RC5

In addition, a MAX232 type RS232 voltage level converter chip
is connected to serial output port RC6.

The program opens a file called MYFILE55.TXT on the SD card
and writes the string "This is MYFILE.TXT." to this file. Then
the string "This is the added data..." is appended to this file.
The program then sends the contents of this file to the UART.

Author:    Dogan Ibrahim
Date:      August 2007
File:      SD3.C
*****************************************************************/

char filename[] = "MYFILE55TXT";
unsigned char txt[] = "This is the added data...";
unsigned short character;
unsigned long file_size,i;

void main()
{

//
// Configure the serial port
//
  Usart_Init(2400);
//
// Initialise the SPI bus
//
   Spi_Init_Advanced(MASTER_OSC_DIV16,DATA_SAMPLE_MIDDLE,
                     CLK_IDLE_LOW, LOW_2_HIGH);
//
// Initialise the SD card bus
//
  while(Mmc_Init(&PORTC,2));
//
// Initialise the FAT file system
//
  while(Mmc_Fat_Init(&PORTC,2));
//
// Create the file (if it doesn't exist)
//
  Mmc_Fat_Assign(&filename,0x80);
//
// Clear the file, start with new data
//
```

图7-12　项目的程序清单

PDG

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

tyw藏书

```
  Mmc_Fat_Rewrite();
//
// Write data to the file
//
  Mmc_Fat_Write("This is MYFILE.TXT.",19);
//
// Add more data to the end...
//
    Mmc_Fat_Append();
    Mmc_Fat_Write(txt,sizeof(txt));
  //
  // Now read the data and send to UART
  //
    Mmc_Fat_Reset(&file_size);
    for(i=0; i<file_size; i++)
    {
       Mmc_Fat_Read(&character);
       Usart_Write(character);
    }

    for(;;);                                    // wait here forever

  }
```

图7-12 （续）

表7-8 Mmc_Fat_Assign文件创建标记

标　记	描　述
0x01	只读
0x02	隐藏
0x04	系统
0x08	容量标签
0x10	子目录
0x20	档案文件
0x40	设备（只在内部使用，在磁盘上找不到）
0x80	文件创建标记。如果文件不存在且此标记已被设置，那么使用指定的名称创建一个新文件

395～396

图7-13给出了运行HyperTerminal后屏幕的快照。

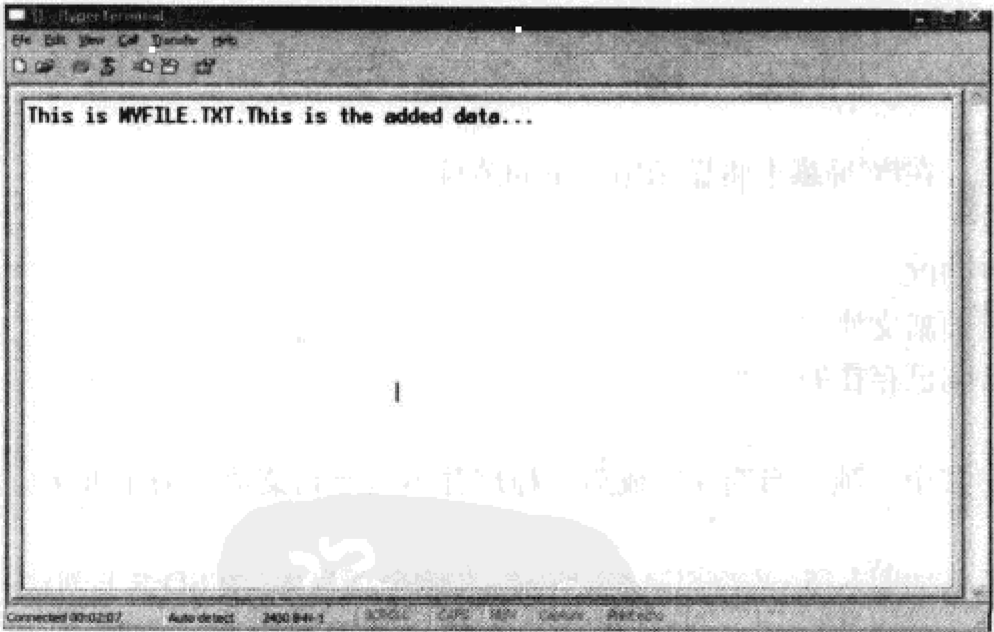


图7-13 屏幕快照

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

lyw藏书

项目7.4　温度记录仪

本项目将展示温度数据记录系统的设计。每10秒钟读取一次周围环境的温度，并存储在SD卡的文件中。本程序是基于菜单的，并且给用户提供了如下的选项：

- 发送保存的文件内容到PC
- 保存读取的温度到SD卡上的一个新文件
- 添加读取的温度到SD卡上的一个已存在的文件

本项目的硬件和项目7.1（如图7-8所示）的很相似，只是这里作为补充，将串行输入端口引脚（RC7）连接到RS232连接器，以接收来自PC键盘的数据。此外，将一个LM35DZ型模拟温度传感器连接到微控制器的模拟输入AN0（引脚2）。新的电路原理图如图7-14所示。

397

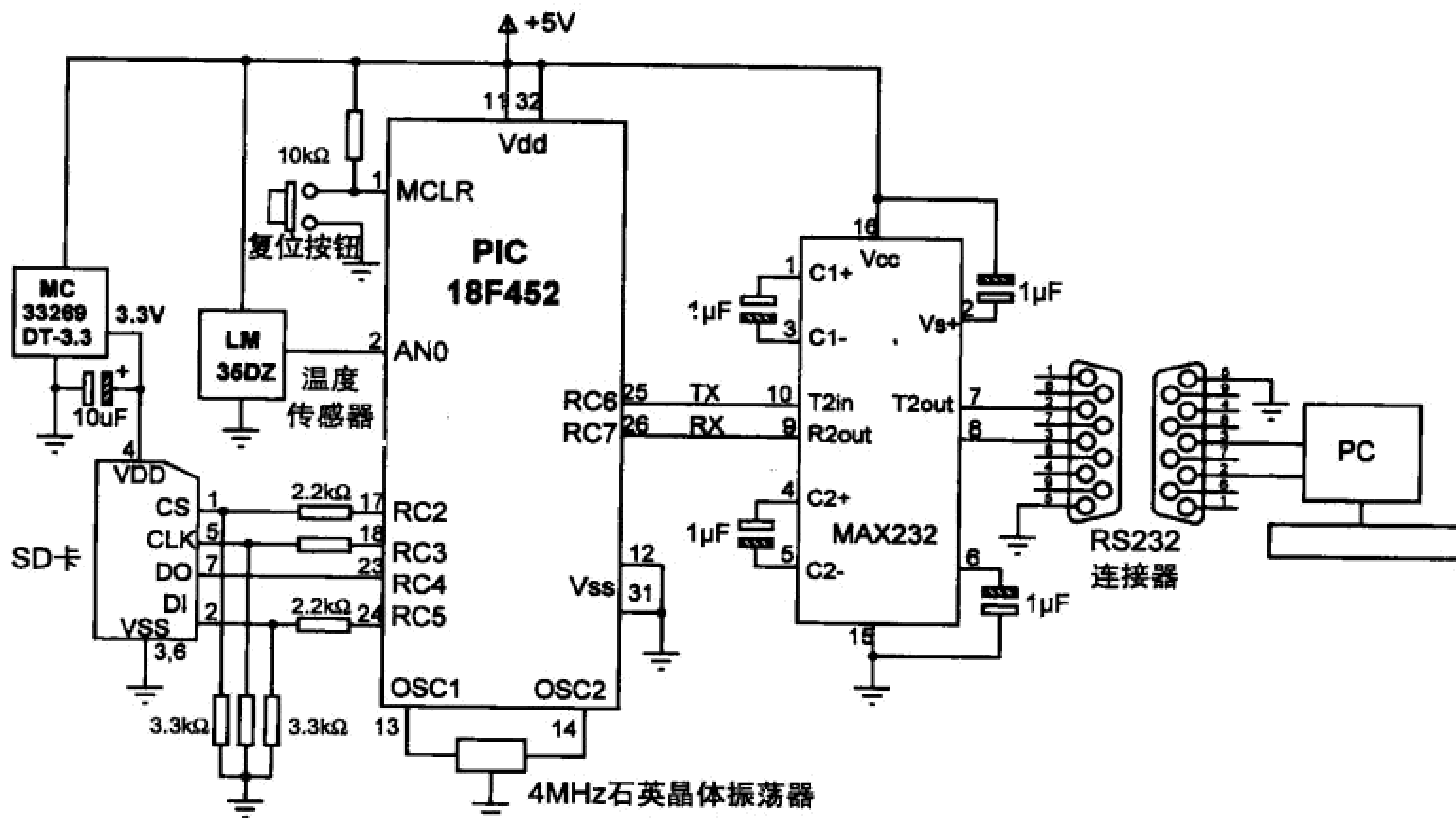


图7-14　项目的电路原理图

LM35DZ是一个三引脚的模拟温度传感器，可以用来测量0℃～+100 ℃之间的温度，精确度可达到1 ℃。该设备的一个引脚连接到电源（+5 V），另一个引脚接地，第三个连接到模拟输出。该传感器的输出电压和温度成正比（如Vo=10 mV/℃）。例如，如果温度是10 ℃，输出电压将是100 mV；如果温度是35 ℃，输出电压将是350 mV。

程序开始运行后，在PC屏幕上将显示出如下的菜单：

温度数据记录仪

1. 发送温度数据到PC
2. 保存温度数据到新文件
3. 添加温度数据到已存在的文件

398

选择哪一个？

然后，用户选择其中一项。当选择完成后，程序并不返回到菜单。为了再次显示菜单，必须重新启动系统。

本项目的程序清单如图7-15（程序SD4.C）所示。在这个项目中，在SD卡上创建一个名为TEMPERTR-TXT的文件（如果不存在的话）来存储读取的温度值（调用库函数，将插入“.”，使得文件名变为“TEMPERTR.TXT”）。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
/*******************************************************************
                  TEMPERATURE LOGGER PROJECT
                  ===========================

In this project a SD card is connected to PORTC as follows:

   CS    RC2
   CLK   RC3
   DO    RC4
   DI    RC5

In addition, a MAX232 type RS232 voltage level converter chip
is connected to serial ports RC6 and RC7. Also, a LM35DZ type
analog temperature sensor is connected to analog input AN0 of
the microcontroller.

The program is menu based. The user is given options of either
to send the saved temperature data to the PC, or to read and
save new data on the SD card, or to read temperature data and
append to the existing file. Temperature is read at every 10
seconds.

The temperature is stored in a file called "TEMPERTR.TXT"

Author:    Dogan Ibrahim
Date:      August 2007
File:      SD4.C
*******************************************************************/

char filename[] = "TEMPERTRTXT";
unsigned short character;
unsigned long file_size,i,rec_size;
unsigned char ch1,ch2,flag,ret_status,choice;
unsigned char temperature[10],txt[12];

//
// This function sends carriage-return and line-feed to USART
//
void Newline()
{
   Usart_Write(0x0D);                // Send carriage-return
   Usart_Write(0x0A);                // Send line-feed
}


//
// This function sends a space character to USART
//
void Space()
{
   Usart_Write(0x20);
}


//
// This function sends a text to serial port
//
void Text_To_Usart(unsigned char *m)
{
   unsigned char i;
```

图7-15 项目的程序清单

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
  i = 0;
  while(m[i] != 0)
  {                                        // Send TEXT to serial port
    Usart_Write(m[i]);
    i++;
  }
}

//
// This function reads the temperature from analog input AN0
//
void Get_Temperature()
{
   unsigned long Vin, Vdec,Vfrac;
   unsigned char op[12];
   unsigned char i,j;

   Vin = Adc_Read(0);                      // Read from channel 0 (AN0)
   Vin = 488*Vin;                          // Scale up the result
   Vin = Vin /10;                          // Convert to temperature in C
   Vdec = Vin / 100;                       // Decimal part
   Vfrac = Vin % 100;                      // Fractional part
   LongToStr(Vdec,op);                     // Convert Vdec to string in "op"
//
// Remove leading blanks
//
     j=0;
     for(i=0;i<=11;i++)
     {
      if(op[i] != ' ')                     // If a blank
      {
        temperature[j]=op[i];
        j++;
      }
     }

     temperature[j] = '.';                 // Add "."
     ch1 = Vfrac / 10;                     // fractional part
     ch2 = Vfrac % 10;
     j++;
     temperature[j] = 48+ch1;              // Add fractional part
     j++;
     temperature[j] = 48+ch2;
     j++;
     temperature[j] = 0x0D;                // Add carriage-return
     j++;
     temperature[j] = 0x0A;                // Add line-feed
     j++;
     temperature[j]='\0';
}



//
// Start of MAIN program
//
void main()
{
  rec_size = 0;
//
```

图7-15 （续）

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
// Configure A/D converter
//
  TRISA = 0xFF;
  ADCON1 = 0x80;                                    // Use AN0, Vref = +5V
//
// Configure the serial port
//
  Usart_Init(2400);
//
// Initialise the SPI bus
//
  Spi_Init_Advanced(MASTER_OSC_DIV16,DATA_SAMPLE_MIDDLE,
                    CLK_IDLE_LOW, LOW_2_HIGH);
//
// Initialise the SD card bus
//
  while(Mmc_Init(&PORTC,2));
//
// Initialise the FAT file system
//
  while(Mmc_Fat_Init(&PORTC,2));
//
// Display the MENU and get user choice
//
  Newline();
  Text_To_Usart("TEMPERATURE DATA LOGGER");
  Newline();
  Newline();
  Text_To_Usart("1. Send temperature data to the PC");
  Newline();
  Text_To_Usart("2. Save temperature data in a new file");
  Newline();
  Text_To_Usart("3. Append temperature data to an existing file");
  Newline();
  Newline();
  Text_To_Usart("Choice ? ");

//
// Read a character from the PC keyboard
//
  flag = 0;
  do {
   if (Usart_Data_Ready())                         // If data received
   {
     choice = Usart_Read();                        // Read the received data
     Usart_Write(choice);                          // Echo received data
     flag = 1;
   }
  } while (!flag);
  Newline();
  Newline();

//
// Now process user choice
//
  switch(choice)
  {
      case '1':
             ret_status = Mmc_Fat_Assign(&filename,1);
             if(!ret_status)
```

图7-15 （续）

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
            {
              Text_To_Usart("File does not exist..No saved data...");
              Newline();
              Text_To_Usart("Restart the program and save data to the file...");
              Newline();
              for(;;);
            }
            else
            {
             //
             // Read the data and send to UART
             //
             Text_To_Usart("Sending saved data to the PC...");
             Newline();
             Mmc_Fat_Reset(&file_size);
             for(i=0; i<file_size; i++)
             {
                 Mmc_Fat_Read(&character);
                 Usart_Write(character);
         }
         Newline();
         text_To_Usart("End of data...");
         Newline();
         for(;;);
        }
case '2':
        //
        // Start the A/D converter, get temperature readings every
        // 10 seconds, and then save in a NEW file
        //
        Text_To_Usart("Saving data in a NEW file...");
        Newline();
        Mmc_Fat_Assign(&filename,0x80);                 // Assign the file
        Mmc_Fat_Rewrite();                              // Clear
        Mmc_Fat_Write("TEMPERATURE DATA - SAVED EVERY 10
                          SECONDS\r\n",43);
        //
        // Read the temperature from A/D converter, format and save
        //
        for(;;)
        {
          Mmc_Fat_Append();
          Get_Temperature();
          Mmc_Fat_Write(temperature,9);
          rec_size++;
          LongToStr(rec_size,txt);
          Newline();
          Text_To_Usart("Saving record:");
          Text_To_Usart(txt);
          Delay_ms(10000);
        }
        break;
case '3':
        //
        // Start the A/D converter, get temperature readings every
        // 10 seconds, and then APPEND to the existing file
        //
        Text_To_Usart("Appending data to the existing file...");
        Newline();
        ret_status = Mmc_Fat_Assign(&filename,1);     // Assign the file
```

图7-15 （续）

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
            if(!ret_status)
            {
              Text_To_Usart("File does not exist - can not append...");
              Newline();
              Text_To_Usart("Restart the program and choose option 2...");
              Newline();
              for(;;);
            }
            else
             {
             //
             // Read the temperature from A/D converter, format and save
             //
                for(;;)
                {
                   Mmc_Fat_Append();
                   Get_Temperature();
                   Mmc_Fat_Write(temperature,9);
                   rec_size++;
                   LongToStr(rec_size,txt);
                   Newline();
                   Text_To_Usart("Appending new record:");
                   Text_To_Usart(txt);
                   Delay_ms(10000);
                }
             }
        default:
             Text_To_Usart("Wrong choice...Restart the program and try again...");
             Newline();
             for(;;);
    }
}
```

图7-15 （续）

在程序的开始，主程序之前，创建如下的函数：

Newline 用来发送回车和换行信号到UART，这样可以将光标移动到下一行。

Text_To_Usart 用来接收一个文本字符串作为它的参变量，并将其发送到UART，在PC屏幕上显示出来。

Get_Temperature 用来开始A/D转换，接收转换后的数据并赋值给变量Vin。对应于这个值的电压以毫伏为单位进行计算，将其除以10得到实际的测量温度（单位是℃）。所得温度值的整数部分通过调用函数LongToStr被转换成字符串形式。清除字符串前面的空白部分，将得到的字符串存储到字符数组temperature中。然后将测量温度值的小数部分、回车和换行添加到字符数组中，该字符数组稍后将被写入到SD卡。

在主程序内，执行以下的操作：

- 将UART的波特率初始化为2 400；
- 初始化SPI总线；
- 初始化FAT文件系统；
- 在PC的屏幕上显示菜单；
- 从用户得到一个选项（1、2还是3）；
- 如果选项是1，则分配温度文件，读取温度记录，并将其显示在PC的屏幕上；
- 如果选项是2，则创建一个新的温度文件，每10秒钟读取一次新的温度值，并将它们存储在文件中；
- 如果选项是3，则分配温度文件，每10秒钟读取一次新的温度值，并将它们添加到已存在的文件中；
- 如果选项不是1、2或者3，则在屏幕上显示错误信息。

399

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

下面将更加详细地描述菜单选项。

选项1。程序将尝试分配已存在的温度文件。如果文件不存在，则在屏幕上显示错误信息“File does not exist...No saved data...”和“Restart the program and save data to the file...”，此时建议用户重新运行程序。另一方面，如果温度文件已经存在，那么在PC的屏幕上将显示信息“Sending saved data to the PC...”。调用函数Mmc_Fat_Reset，将文件指针指向文件开始处，并且返回文件大小，以字节为单位。接下来，构造一个for循环，每次调用函数Mmc_Fat_Read从卡中读取一个字节的温度数据，然后调用函数Mmc_Usart_Write，将这些记录数据发送到PC的屏幕上。在数据的结尾，信息“End of data...”被发送到PC的屏幕上。

400～405

选项2。在这个选项中，将信息“Saving data in a NEW file...”发送到PC的屏幕上，并调用函数Mmc_Fat_Assign，使用创建标记0x80，来创建一个新的文件。调用函数Mmc_Fat_Write，将信息“TEMPERATURE DATA-SAVED EVERY 10 SECONDS”写在文件的第一行。然后，构造一个for循环，调用函数Mmc_Fat_Append，将SD卡设置为文件添加模式，然后调用函数Get_Temperature读取新的温度数据。随后，将温度值写入SD卡。此外，当前记录编号显示在PC的屏幕上，用来表明程序确实在运行。这个过程在延时10秒钟后又会重复。

选项3。这个选项很像选项2，唯一不同的是，它并没有创建新文件，而是以读取模式打开已存在的温度文件。如果文件不存在，那么将在PC屏幕上显示错误信息。

其他。如果用户输入的是1、2、3以外的数字，那么将执行这个选项，并在PC的屏幕上显示错误信息“Wrong choice...Restart the program and try again...”。

将微控制器的输出端连接到PC的串行端口（如COM1），然后运行HyperTerminal终端模拟软件来测试该项目。设置通信参数为2 400的波特率、8个数据位、1个停止位、无奇偶位。图7-16给出了选择选项2在新文件中保存温度记录时的PC屏幕快照。注意到，在写入SD卡时，当前记录编号也显示在PC的屏幕上。

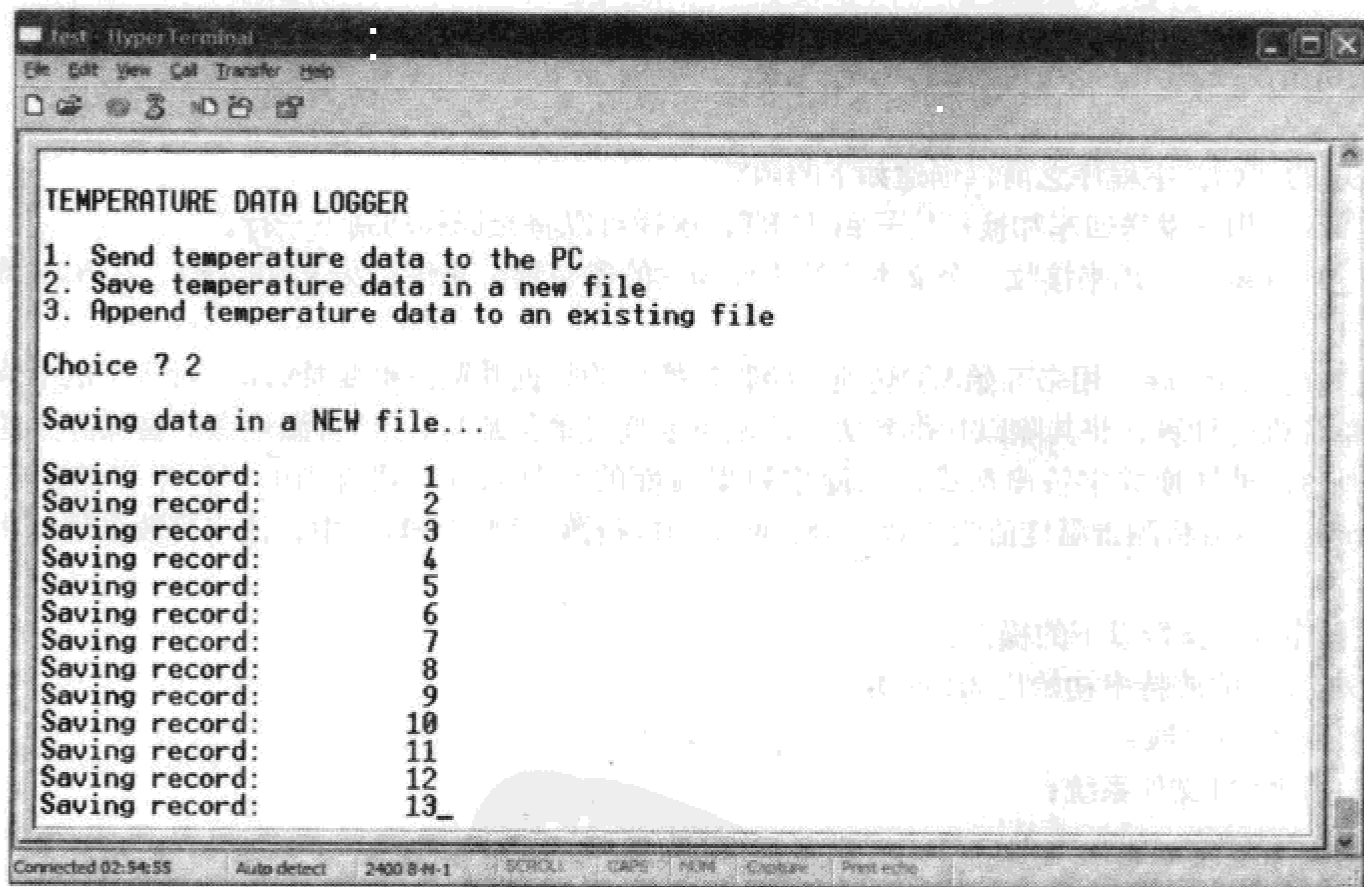


图7-16　使用选项2，在SD卡上保存温度记录

406

图7-17给出了选择选项1从SD卡中读取温度记录并将它们显示在PC屏幕上时的屏幕快照。

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

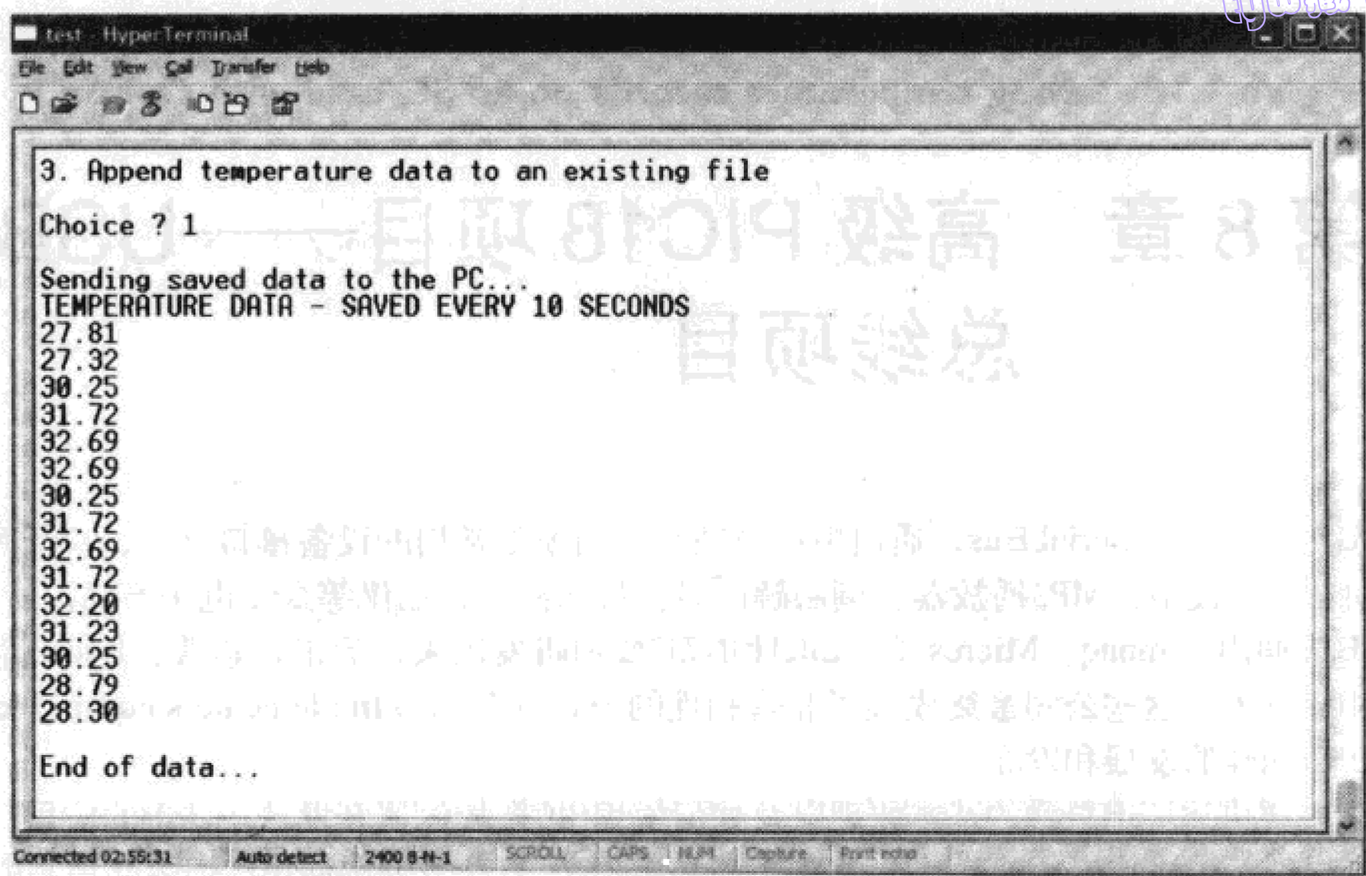


图7-17　使用选项1，在PC屏幕上显示记录

407

最后，图7-18给出了选择选项3将温度读取值添加到已存在的文件时的屏幕截图。

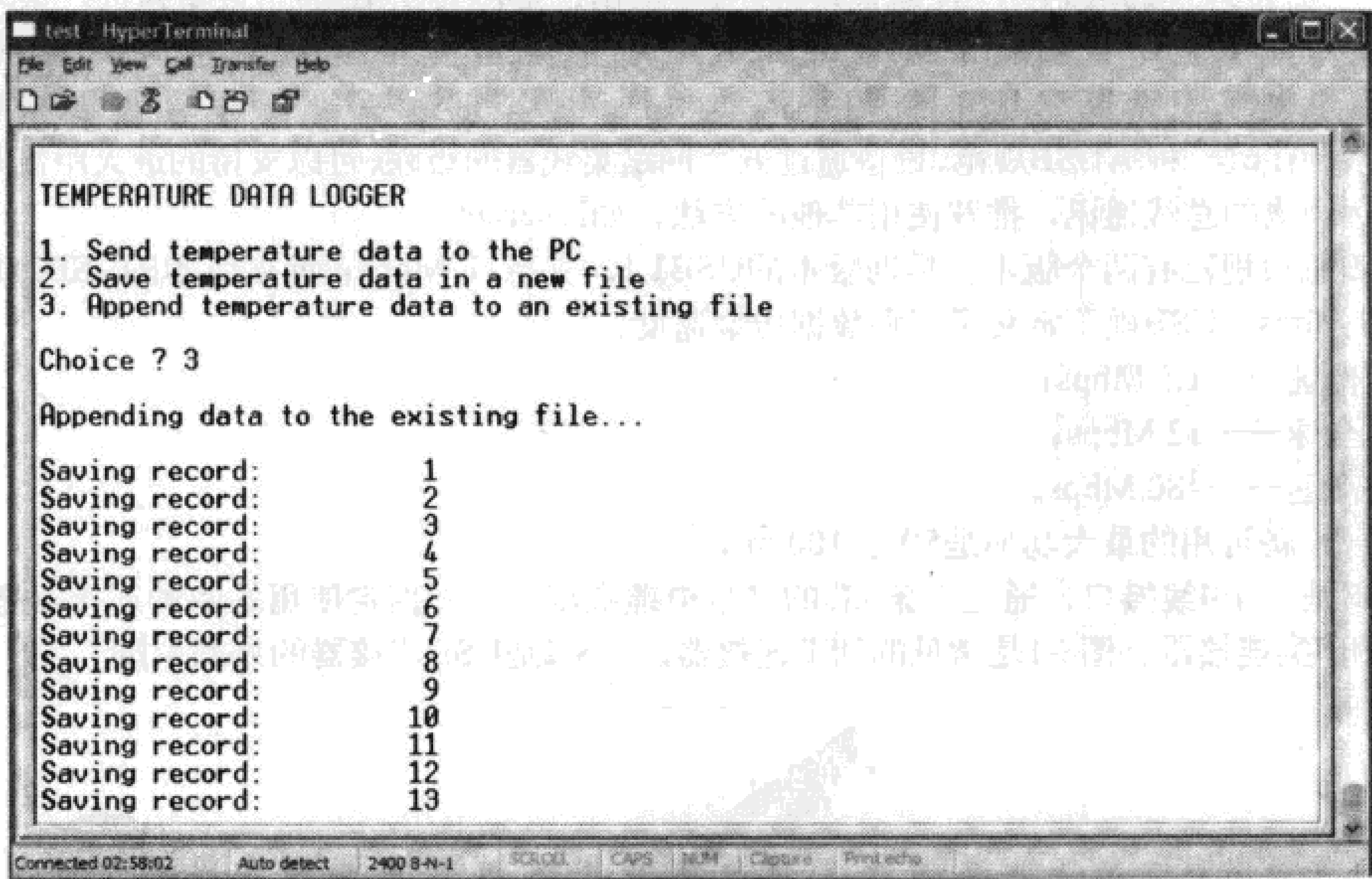


图7-18　使用选项3，在SD卡上保存温度记录

408



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

第 8 章　高级 PIC18 项目——USB 总线项目

USB（Universal Serial Bus，通用串行总线）是当今最常用的设备接口之一，广泛用于PC、数码相机、GPS设备、MP3播放器、调制解调器、打印机、扫描仪等众多电子产品。

USB最初由Compaq、Microsoft、Intel和NEC公司研发出来，后来，惠普、Lucent和Philips公司也相继加入。这些公司最终成立了非营利性的公司（即USB Implementers Forum Inc.）来专门组织USB标准的发展和发布。

本章主要讲述USB总线的基本原理以及如何使用PIC微控制器开发基于USB的应用。USB总线是一个很复杂的协议。本章不讨论USB的设计和使用，只是罗列出使用USB需要用到的基本原理。此外，还涉及mikroC语言提供的一些函数，因其简化了基于USB的微处理器项目的设计。

USB是一种高速串行接口，能够向与之相连接的设备提供电源。USB总线支持多达127个设备（这受到7位地址的限制——注意地址0没有被使用，留作其他用途），设备通过一条3m～5m长的4线串行电缆与USB连接。很多USB设备可以通过集线器与同一总线相连。集线器分4端口、8端口甚至16端口。设备可以连接到集线器，集线器又可以连接到另一个网路集线器，但级联的最大数限制在6层。根据USB规范，设备通过五个网路集线器的级联，可以支持的最大距离是30 m。如需更远距离的总线通讯，推荐使用其他的方法，如Ethernet。

409

USB总线规范有两个版本：早期版本的USB1.1，支持11 Mbps；而新版本的USB2.0，支持高达480 Mbps。USB规范定义了三种数据传输速度：

- 低速——1.5 Mbps；
- 全速——12 Mbps；
- 高速——480 Mbps。

外部设备可用的最大功率是5 V、100 mA。

USB是一个4线接口，通过一条4芯的屏蔽电缆实现。USB指定使用的两种连接器是：A类连接器和B类连接器。图8-1是常见的USB连接器。图8-2是USB连接器的外部引脚。

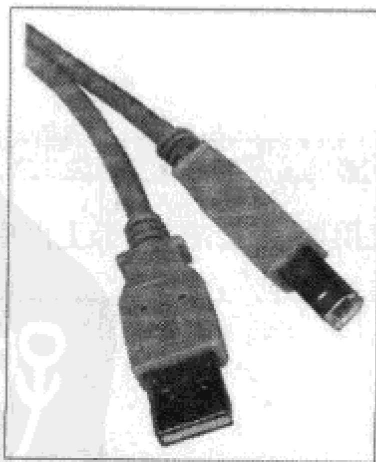


图8-1　USB连接器

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

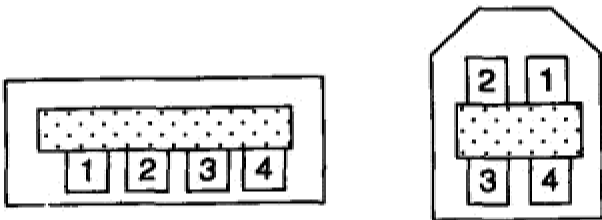


图8-2　USB连接器的外部引脚

信号线的颜色是指定的。表8-1给出了A类连接器和B类连接器的引脚和信号线颜色。 410

表8-1　USB连接器的引脚分配

引脚编号	名　　称	颜　　色
1	+5.0V	红
2	Data−	白
3	Data+	绿
4	接地	黑

USB规范也定义了微型的总线连接器，这种连接器主要用于便携式的电子设备中，如数码相机和其他的手持设备。该连接器有第5个引脚（即ID），尽管此引脚未被使用。微型总线连接器的引脚参数和信号线颜色如表8-2所示。

表8-2　微型总线连接器的引脚参数

引脚编号	名　　称	颜　　色
1	+5.0V	红
2	−Data	白
3	+ Data	绿
4	未被使用	—
5	接地	黑

在这些引脚中，两个引脚Data+和Data−构成了双绞线，可传输差动数据信号和一些单端数据状态。 411

USB信号是双相的，由主机发送的信号使用NRZI（不归零就反向）数据编码技术。在这种技术下，当信号变为逻辑0时，信号电平反向；当信号为逻辑1时，信号电平保持不变。在数据流中，每6个连续位后将补充一个0，这使得数据是动态的（这叫补位，因为附加位延长了数据流）。图8-3说明了NRZI是如何实现的。

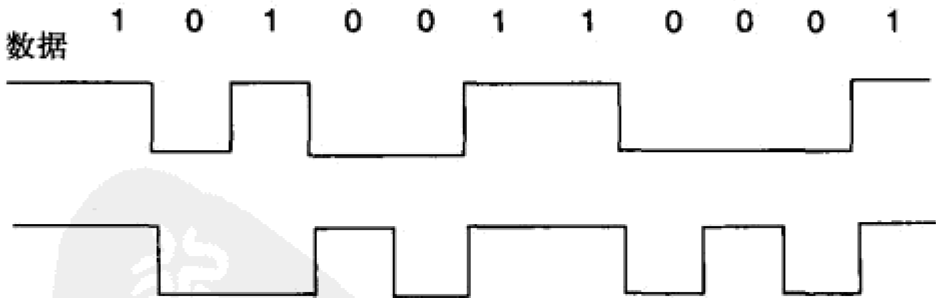
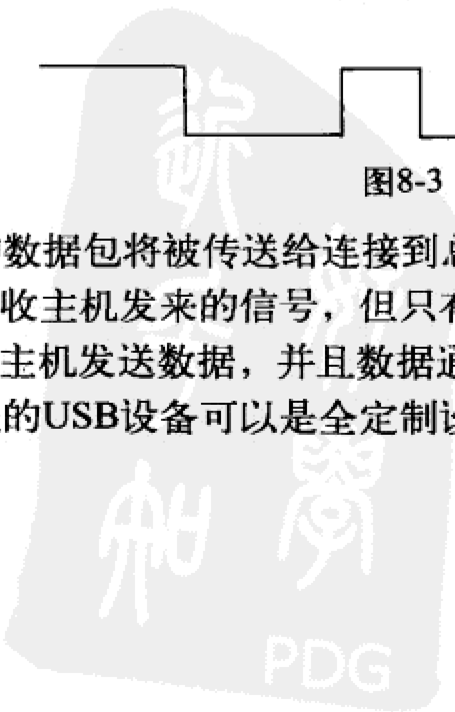


图8-3　NRZI数据

由主机发送的数据包将被传送给连接到总线的每一个设备，并通过网路集线器一直往下传送。所有设备都接收主机发来的信号，但只有地址吻合的设备才接收数据。反过来说，每一次只能有一个设备向主机发送数据，并且数据通过集线器一级一级地往上传送，直至到达主机。

连接到总线上的USB设备可以是全定制设备，这需要一个全定制的设备驱动器，或者它们



世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

属于同一类设备。设备类使得具有相似功能的设备系列可以共享相同的设备驱动器。例如，一个打印机的设备类是0x07，大部分打印机都使用这类驱动器。

最常见的设备类如表8-3所示。USB人机接口设备（HID）类非常有趣，本章的项目将使用到它。

表8-3　USB设备类

设备类	描　述	设备举例
0x00	保留	—
0x01	USB音频设备	声卡
0x02	USB通信设备	调制解调器、传真
0x03	USB人机接口设备	键盘、鼠标
0x07	USB打印机设备	打印机
0x08	USB大容量存储设备	内存卡、闪存
0x09	USB集线器设备	集线器
0x0B	USB智能卡阅读器设备	读卡器
0x0E	USB视频设备	网络相机、扫描仪
0xE0	USB无线设备	蓝牙

以下是一些常见的USB术语。

终端（Endpoint）：终端是数据的源端或接收器，一个USB设备可以有很多的终端，至多16个输入终端和16个输出终端。

412

事务处理（Transaction）：事务处理是指总线上的数据传输。

通道（Pipe）：通道是指主机与终端之间的逻辑数据连接。

8.1　总线速度识别

在总线的设备端，D+或D−线经一个1.5 kΩ的上拉电阻连接到3.3 V电源。对于全速总线，D+线经上拉电阻连接到3.3 V电源，而对于低速总线，D−经上拉电阻连接到3.3 V电源。若没有设备连接到总线时，则主机将两条数据线都看作低电平。一旦有设备连接到总线，就会把D+或D−线的电位拉到高电平，主机据此可以判断已有设备连接到总线。通过观察出现高电平的数据线，可以判断设备的传输速度。

8.2　USB 状态

USB总线的部分状态如下。

空闲（IDLE）状态：当上拉线为高电平而其他线为低电平时，总线处于空闲状态，这是在数据包发送前和发送后的总线状态。

未连接（Detached）状态：当没有设备连接到总线上时，两数据线均为低电平。

连接（Attached）状态：当有设备连接到总线时，主机检测到D+或D−变为高电平，这表明已有设备接入。

413

J状态（J state）：如同空闲状态。

K状态（K state）：与J状态相反的状态。

单端0（Single-Ended Zero，SE0）状态：在单端0状态时，总线的两条线都为低电平。

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

tyw藏书

单端1（Single-Ended One，SE1）状态：在单端1状态时，总线的两条线都为高电平；单端1状态是非法的，应该加以避免。

复位（Reset）状态：当主机要与总线上的设备进行通信时，需要先发送一个Reset信号，这可以通过将两数据线拉至低电平并保持至少10 ms来实现。

EOP状态：这是包的结束状态，基本上是由一个2位时间的SE0状态和紧接着的一个1位时间的J状态组成。

保持激活（Keep alive）状态：这是EOP得到的状态；每毫秒至少发送一次保持激活信号，以保证设备不被悬挂。

悬挂（Suspend）状态：该状态用来节约能耗；保持3 ms不向任何设备发送信息，设备将进入悬挂状态；一个处于悬挂状态的设备从总线获得的电流不超过0.5 mA，并且必须能识别复位信号和恢复信号。

恢复（Resume）状态：改变数据线上信号的极性并持续至少20 ms，紧接着发送一个低速EOP信号，此时，处于悬挂状态的设备被激活。

8.3 USB总线通信

USB是一个以主机为中心的连接系统，由主机控制USB的使用。连接到总线上的设备都分配有一个唯一的地址，在没有得到主机许可时，设备不能发送总线信号。当有新的USB设备连接到总线时，USB主机使用地址0来访问设备的基本信息。然后，主机将为设备分配一个唯一的USB地址。在主机请求并接收到设备的更多信息（如制造商名称、设备性能、产品ID）后，就可以开始双向通信了。

8.3.1 数据包

在USB总线上，数据以包的形式传送。作为开始，包使用一个同步信号来允许接收器时钟同步数据的传输，紧接着是包的数据字节，最后以一个包结束信号作为结尾。 414

在每一个USB信息的同步字段后紧随一个包标识符（PID）字节。PID本身占4个位，余下的4位是前4位的补码。这里有17个不同的PID值（如表8-4所示），其中包括1个保留值和一个表示两种不同含义的复用值。

表8-4 PID值

PID类型	PID名称	二进制位	描述
标记	OUT	1110 0001	从主机到设备的传送
	IN	0110 1001	从设备到主机的传送
	SOF	1010 0101	帧的开始
	SETUP	0010 1101	设置命令
数据	DATA0	1100 0011	数据包PID（偶数字节）
	DATA1	0100 1011	数据包PID（奇数字节）
	DATA2	1000 0111	数据包PID（高速）
	MDATA	0000 1111	数据包PID（高速）
握手	ACK	1101 0010	接收器接受包
	NAK	0101 1010	接收器不接受包
	STALL	0001 1110	暂停
	NYET	1001 0110	接收器未应答

PDG

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

tyw藏书

（续）

PID类型	PID名称	二进制位	描 述
特殊功能	PRE	0011 1100	主机同步码
	ERR	0011 1100	数据分割传输错误
	SPLIT	0111 1000	高速数据分割传输
	PING	1011 0100	高速流控制
	Reserved	1111 0000	保留

根据包开始处的不同的PID，可以分为四种包格式：标记包、数据包、握手包和特殊功能包。

图8-4给出了标记包的格式，该包用于OUT、IN、SOF（帧的开始）和SETUP。该包由一个7位的地址、一个4位的ENDP（终端号）、一个5位的CRC校验和以及一个EOP（包的结尾）组成。

同步信号	PID	ADDR	ENDP	CRC	EOP
	8B	7B	4B	5B	

图8-4 标记包

数据包用于DATA0、DATA1、DATA2和MDATA数据传输。数据包格式如图8-5所示，由PID、0～1 024 B的数据、一个2 B的CRC校验和，以及一个EOP组成。

415

同步信号	PID	数据	CRC	EOP
	1 B	0~1024 B	2 B	

图8-5 数据包

同步信号	PID	EOP
	1 B	

图8-6 握手包

图8-6为握手包的格式，用于ACK、NAK、STALL和NYET。当接收器确定已接收到一个无误的数据包时，就发送ACK；当接收器不能接收到数据包时，就发送NAK；当终端处在暂停状态时，就发送STALL；而当没有来自接收器的应答信号时，就发送NYET。

8.3.2 数据流类型

在USB总线上，数据传输有四种方式：块传输、中断传输、同步传输和控制传输。

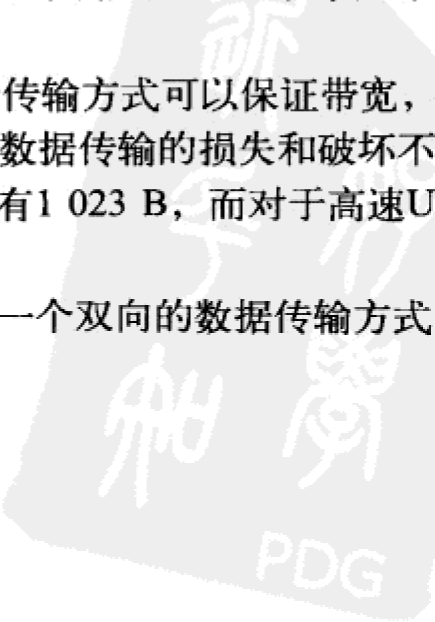
块传输 当要求无误传输且带宽无法保证时，大量数据的传输可以采用块传输方式。如果将一个OUT终端定义为使用块传输，那么主机将会按照OUT设置向终端传输数据。类似地，如果将一个IN终端定义为使用块传输，那么主机就会按照IN设置从终端接收数据。一般而言，如果传输的低速率不是问题的话，可以使用块传输方式。在块传输方式下，对于全速USB，最多可传输8～64个包；而对于高速USB则可达512个包（块传输不允许工作在低速USB下）。

中断传输 当要传输的数据量小且带宽较高时，可以使用中断传输方式。在高的带宽下，数据必须尽可能无延迟地快速传输。注意到，中断传输与电脑系统里的中断无关。对于低速USB，中断包的大小可以是1 B～8 B不等；对于全速USB可以是1 B～64 B不等；对于高速USB则可达1 024 B。

416

同步传输 同步传输方式可以保证带宽，但不能保证无误地传输。这种传输方式通常在对速度要求比较高而对数据传输的损失和破坏不是很关心的场合，如音频数据的传输。对于全速USB，同步传输包可有1 023 B，而对于高速USB，则可达1 024 B。（同步传输不允许工作在低速USB下）。

控制传输 这是一个双向的数据传输方式，可同时使用IN终端和OUT终端。控制传输通常



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

被主机用来对设备进行初始化。对于低速USB，控制传输包最多可包含8B；对于全速USB，可以是8B～64B不等；而对于高速USB，则可达64B。控制传输的执行分三个部分：SETUP、DATA和STATUS。

8.3.3 枚举

当设备连接到USB总线时，主机通过枚举过程可以发现设备。枚举的过程如下。

- 当有设备接入时，主机可以发现它，是因为数据线D+或D-被拉成高电平了。
- 主机发送一个USB复位信号给设备，使设备处于已知状态。复位设备对地址0作出应答。
- 主机使用获取描述符（Get Descriptor）指令向设备发送地址0请求，以得到设备包的最大值。
- 设备应答，发送一小部分的描述符。
- 主机再次发送一个复位信号。
- 主机给设备分配一个唯一的地址，并向设备发送一个设置地址（Set Address）请求。在请求完成后，设备得到了新的地址。在这一步，主机可以对总线上的其他任何新接入的设备进行复位。
- 主机发送获取设备描述符（Get Device Descriptor）请求以得到完整的设备描述符，收集制造商、设备类型和最大的控制包等信息。
- 主机发送获取配置描述符（Get Configuration Descriptor）请求以得到设备的配置数据，如电源要求以及支持的接口类型和数量。
- 主机还可以请求获取设备的其他任何描述符。

主机和设备之间的初始通信是通过控制传输方式实现的。

417

起初，设备是有地址的，但是处于未配置的状态。在主机收集了设备的足够信息之后，主机向设备发送一个设置配置（Set Configuration）请求来加载一个合适的设备驱动器，用以配置设备。此时，设备配置完成，已做好准备应答来自主机的设备请求（如发送数据给主机，或者接收来自主机的数据）。

8.4 描述符

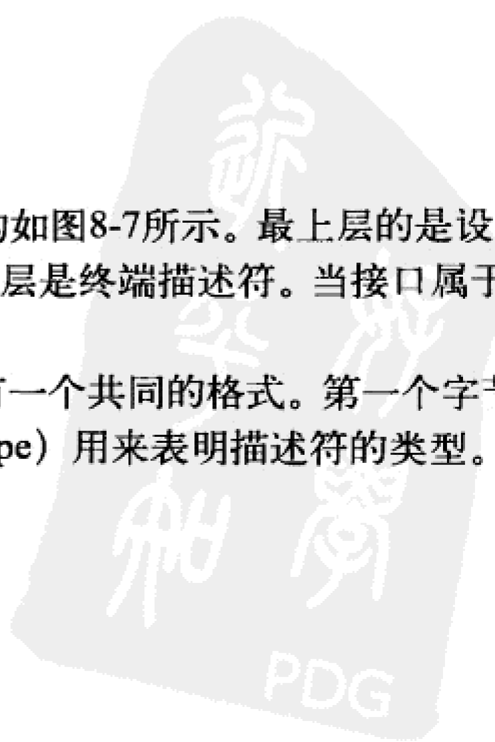
所有USB设备都有不同层次的描述符，用来描述设备的不同特性：如制造商ID、设备版本、支持的USB版本、设备类型、电源要求、终端类型和数量等。

最常用的USB描述符有：

- 设备描述符
- 配置描述符
- 接口描述符
- HID描述符
- 终端描述符

描述符的层次结构如图8-7所示。最上层的是设备描述符，第二层是配置描述符，第三层是接口描述符，最下面一层是终端描述符。当接口属于HID类时，HID描述符总是跟在接口描述符之后。

所有的描述符都有一个共同的格式。第一个字节（bLength）用来指定描述符的长度，第二个字节（bDescriptorType）用来表明描述符的类型。



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

8.4.1　设备描述符

418

设备描述符是来自设备的最上层信息集，这是主机试图从设备读取的第一项。

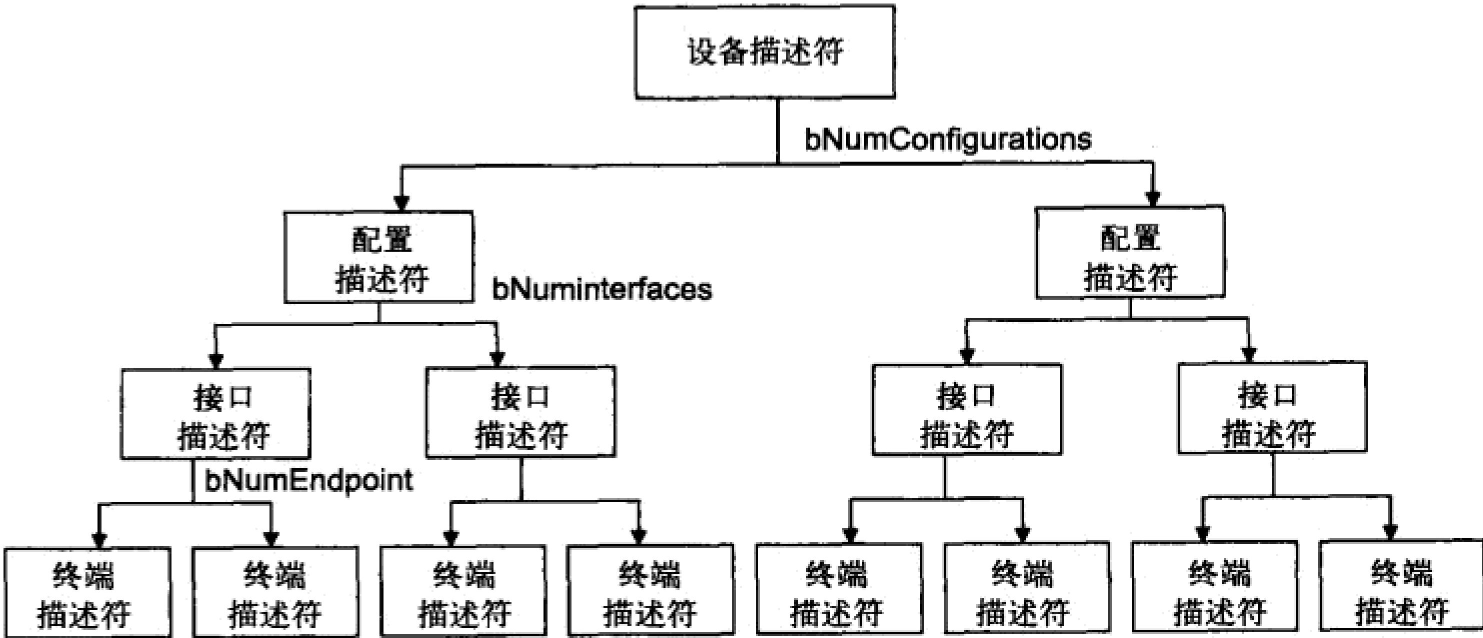


图8-7　USB描述符的层次结构

一个USB设备只有一个设备描述符，因为设备描述符代表了整个设备（图8-7）。它提供了设备的常用信息，如制造商、序列号、产品号、设备类别和配置数据等。表8-5给出了设备描述符的格式以及每个字段的含义。

bLength设备描述符的长度。

bDescriptorType 设备描述符的类型。

bcdUSB以BCD码格式报告设备支持的USB最高版本。该数可表示为0xJJMN，其中JJ是主要版本号，M是次要版本号，N是次次要版本号。例如，USB1.1可表示为0x0110。

bDeviceClass，bDeviceSubClass, bDeviceProtocol这三个字段由USB组织结构分配，可被系统用来发现设备的驱动器。

bMaxPacketSize0终端0的输入输出包的最大值。

idVendor这是供应商ID，由USB组织分配得到。

idProduct这是产品ID，由制造商进行分配得到。

419

bcdDevice这是发布的设备版本号，其格式与bcdUSB相同。

iManufacturer，iProduct，iSerialNumber这是关于制造商和产品的具体信息。这几个字段没有要求，也可以设置为0。

bNumConfigurations设备支持的配置数量。

表8-5　设备描述符

偏移量	字　段	大　小	描　述
0	bLength	1	描述符大小（用字节表示）
1	bDescriptorType	1	设备描述符类型（0x01）
2	bcdUSB	2	支持的USB最高版本
4	bDeviceClass	1	类别号
5	bDeviceSubClass	1	子类别号
6	bDeviceProtocal	1	协议号

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

（续）

偏移量	字段	大小	描述
7	bMaxPacketSize0	1	包的最大值
8	idVendor	2	供应商ID
10	idProduct	2	产品ID
12	bcdDevice	2	发布的设备版本号
14	iManufacturer	1	制造商字符串描述符
15	iProduct	1	产品字符串描述符索引
16	iSerialNumber	1	序列号描述符索引
17	bNumConfigurations	1	可配置的数量

表8-6所示为鼠标设备描述符的例子。描述符的长度为18 B（bLength＝18），描述符类型为0x01（bDescriptorType = 0x01）。设备支持USB1.1（bcdUSB=0x0110）。bDeviceClass，bDeviceSubClass，bDeviceProtocal设置为0，表明设备类型信息在接口描述符中。bMaxPacketSize0设置为8，表明终端0输入输出包大小为0 B～8 B。接下来的三个字节是供应商ID、产品ID、设备版本号。然后，使用三项来定义了制造商、产品和序列号的索引。最后，可以看到，鼠标设备只有一个配置（bNumConfigurations＝1）。 420

表8-6 设备描述符例子

偏移量	字段	值	描述
0	bLength	18	描述符大小为18B
1	bDescriptorType	0x01	设备描述符类型
2	bcdUSB	0x0110	支持的最高USB为USB1.1
4	bDeviceClass	0x00	类别信息在接口描述符
5	bDeviceSubClass	0x00	类别信息在接口描述符
6	bDeviceProtocal	0x00	类别信息在接口描述符
7	bMaxPacketSize0	8	包的最大值
8	idVendor	0x02A	XYZ 有限公司
10	idProduct	0x1001	鼠标
12	bcdDevice	0x0011	发布的设备版本号
14	iManufacturer	0x20	制造商字符串描述符索引
15	iProduct	0x21	产品字符串描述符索引
16	iSerialNumber	0x22	序列号描述符索引
17	bNumConfigurations	1	可配置的数量

8.4.2 配置描述符

配置描述符用来提供诸如电源要求和支持接口的数量及类型等信息。一个设备可以有多个的配置。

表8-7所示为配置描述符的格式，每个字段都有自己特定的含义。

bLength设备描述符的长度。

bDescriptorType 设备描述符的类型。

wTotalLength所有描述符的总长度（即配置描述符+接口描述符+HID描述符+终端描述符的总和）。当主机读取配置描述符时，设备将返回完整的配置信息，包括所有接口描述符和终端描述符。 421

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

tyw藏书

bNumInterfaces配置所需的接口数量。

bConfigurationValue主机使用它来选择配置（使用SetConfiguration指令）。

iConfiguration这是字符串描述符的索引，以可读格式描述配置。

bmAttributes用来描述电源要求。如果设备是采用USB总线供电的，那么需要对D7位进行置位。如果是自供电的，那么需要将D6位进行置位。D5位用来指定设备的远程唤醒功能。D7位、D0～D4位均为保留位。

bMaxPower用来描述设备从总线上获得的最大电流,以2 mA为一个单位进行度量。

表8-7 配置描述符

偏移量	字段	大小	描述
0	bLength	1	描述符大小（用字节表示）
1	bDescriptorType	1	设备描述符类型（0x02）
2	wTotalLength	2	返回字节总数
4	bNumInterfaces	1	接口数量
5	bConfigurationValue	1	用来选择配置的值
6	iConfiguration	1	描述配置字符串的索引
7	bmAttributes	1	电源属性
8	bMaxPower	2	最大功率消耗（以2mA为单位）

表8-8所示为鼠标设备的配置描述符例子。描述符长度为9 B（bLength=9），描述符类型为0x02（bDescriptorType=0x02），所有描述符的总长度为34 B（wTotalLength=34），鼠标的接口数为1（bNumInterfaces=1），在SetConfiguration()函数中，主机SetConfiguration指令必须使用1作为参数。这里没有描述配置的字符串。bmAttributes设置为0x40，表明设备是自供电的。bMaxPower设置为10，说明设备从总线获得的最大电流为20 mA。

422

表8-8 配置描述符例子

偏移量	字段	值	描述
0	bLength	9	描述符大小为9 B
1	bDescriptorType	0x02	设备描述符为0x02
2	wTotalLength	34	返回总字节为34
4	bNumInterfaces	1	接口数量为1
5	bConfigurationValue	1	用来选择配置的值
6	iConfiguration	0x2A	描述配置字符串的索引
7	bmAttributes	0x40	电源属性
8	bMaxPower	10	最大功率消耗为20 mA

8.4.3 接口描述符

接口描述符用来指定接口的类别和使用终端的数量。这里可能使用多个接口。

423

表8-9给出了接口描述符的格式以及各字段的含义。

表8-9 接口描述符

偏移量	字段	大小	描述
0	bLength	1	描述符大小（以字节为单位）
1	bDescriptorType	1	设备描述符（0x04）

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

lyw藏书

（续）

偏移量	字 段	大 小	描 述
2	bInterfaceNumber	1	接口数量
3	bAlternateSetting	1	选择备选设置的值
4	bNumEendpoints	1	终端数量
5	bInterfaceClass	1	类别号
6	bInterfaceSubClass	1	子类别号
7	bInterfaceProtocol	1	协议号
8	iInterface	1	接口字符串描述符索引

bLength这是设备描述符的长度。

bDescriptorType这是设备描述符的类型。

bInterfaceNumber用来表示接口描述符索引。

bAlternateSetting用来指定可由主机使用 Set Interface 指令来选择的备选接口。

bNumEendpoints用来表示接口所使用的终端数。

bInterfaceClass用来指定设备的类别号（由USB组织进行分配得到）。

bInterfaceSubClass用来指定设备的子类别号（由USB组织进行分配得到）。

bInterfaceProtocol用来指定设备的协议号（由USB组织进行分配得到）。

iInterface这是接口的字符串描述符的索引。

表8-10给出的是鼠标设备的接口描述符例子。描述符长度为9 B（bLength=9），描述符类型为0x04 （bDescriptorType=0x04）。用于参考该接口的接口数是1（bInterfaceNumber =1）。 424

表8-10 接口描述符例子

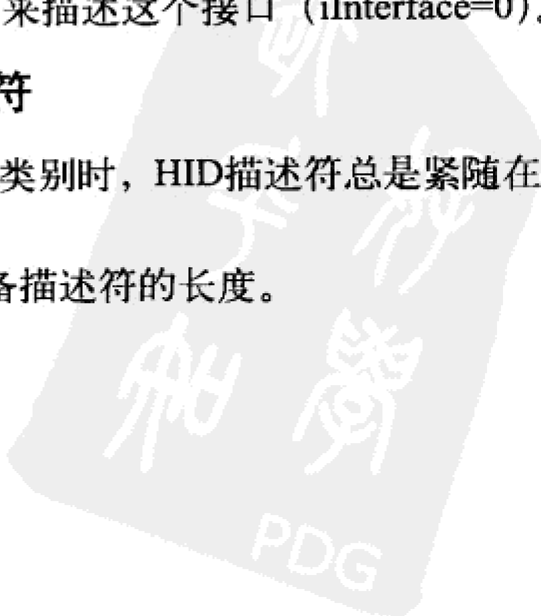
偏移量	字 段	值	描 述
0	bLength	9	描述符大小为9B
1	bDescriptorType	0x04	设备描述符类型（0x04）
2	bInterfaceNumber	0	接口数量
3	bAlternateSetting	0	用来选择备选设置的值
4	bNumEendpoints	1	终端的数量为1
5	bInterfaceClass	0x03	类别号0x03
6	bInterfaceSubClass	0x02	子类别号0x02
7	bInterfaceProtocol	0x02	协议号0x02
8	iInterface	0	接口字符串描述符的索引

bAlternateSetting设置为0（没有可供选择的接口）。这个接口所使用的终端数为1（终端0除外），而且这是鼠标用来发送数据的终端。设备类别号是0x03（bInterfaceClass=0x03）。这是一个HID（人机接口设备）类型的设备。接口子类别号设置为0x02。设备协议号为0x02（鼠标）。最后，没有字符串用来描述这个接口（iInterface=0）。

8.4.4 HID 描述符

当接口属于HID类别时，HID描述符总是紧随在接口描述符之后。表8-11所示为HID描述符的格式。

bLength这是设备描述符的长度。



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

lyw藏书

bDescriptorType这是设备描述符的类型。
bcdHID这是HID类别的说明。
bCountryCode用来指明一些局部的差异。
bNumDescriptors用来表明这个系列是否有附加描述符。
bDescriptorType这是bNumDescriptors中指明的附加描述符的类型。
wDescriptorLength这是附加描述符的长度，以字节为单位。

表8-11　HID描述符

偏移量	字　段	大　小	描　述
0	bLength	1	描述符大小（用字节表示）
1	bDescriptorType	1	HID（0x21）
2	bcdHID	2	HID类别
4	bCountryCode	1	国家（地区）代码
5	bNumDescriptors	1	附加描述符的数量
6	bDescriptorType	1	附加描述符的类型
7	wDescriptorLength	2	附加描述符的长度

425

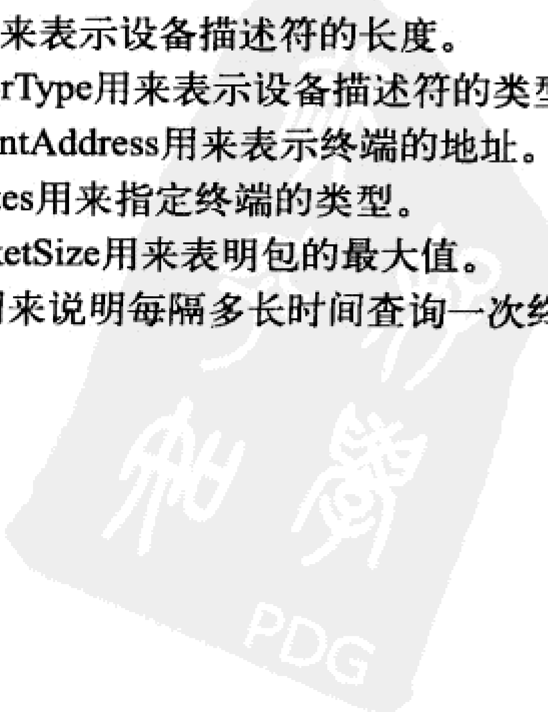
表8-12所示为鼠标设备的HID描述符举例。描述符长度为9 B（bLength=9），描述符类型为0x21（bDescriptorType=0x21）。HID类别设置为1.1（bcdHID=0x0110）。国家代码设置为0（bCountryCode=0），表明此设备没有特别的区域局限。描述符数量设置为1（bNumDescriptors=1），表明这个类别只有一个附加描述符。该附加描述符的类型为REPORT（bDescriptorType=REPORT），长度为52 B（wDescriptorLength=52）。

表8-12　HID描述符例子

偏移量	字　段	大　小	描　述
0	bLength	9	描述符大小为9 B
1	bDescriptorType	（0x21）	HID（0x21）
2	bcdHID	0x0110	类别版本1.1
4	bCountryCode	0	没有特殊的国家代码
5	bNumDescriptors	1	附加描述符的数量
6	bDescriptorType	REPORT	附加描述符的类型
7	wDescriptorLength	5	附加描述符的长度

8.4.5　终端描述符

表8-13给出了终端描述符的格式。
bLength用来表示设备描述符的长度。
bDescriptorType用来表示设备描述符的类型。
bcdEndpointAddress用来表示终端的地址。
bmAttributes用来指定终端的类型。
wMaxPacketSize用来表明包的最大值。
bInterval用来说明每隔多长时间查询一次终端（以ms为单位）。



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

表8-13 终端描述符

偏移量	字段	大小	描述
0	bLength	1	描述符大小（用字节表示）
1	bDescriptorType	1	描述符类型（0x05）
2	bcdEndpointAddress	1	终端的地址
4	bmAttributes	1	终端的类型
5	wMaxPacketSize	2	包的最大值
6	bInterval	1	查询间隔

表8-14所示为鼠标设备的终端描述符例子。描述符长度为7 B（bLength=7），描述符类型为0x05（bDescriptorType=0x05）。终端地址为0x50（bcdEndpointAddress=0x50），该终端被用作中断终端（bmAttributes=0x03）。包的最大值设置为2（wMaxPacketSize=0x02），表明长度大于2 B的包将不能从该终端发送。每隔20ms至少查询终端一次（bInterval=0x14）。 426

表8-14 终端描述符例子

偏移量	字段	大小	描述
0	bLength	7	描述符大小（用字节表示）
1	bDescriptorType	0x05	描述符类型（0x05）
2	bcdEndpointAddress	0x50	终端地址
4	bmAttributes	0x03	中断类型的终端
5	wMaxPacketSize	0x0002	包的最大值为2 B
6	bInterval	0x14	查询间隔为20 ms

8.5 PIC18 微控制器的 USB 总线接口

有一些PIC18微控制器直接支持USB接口。例如，PIC18F4550微控制器包含有一个全速和低速的兼容USB接口，这允许在PC和微控制器之间进行通信。在本章的USB项目中，将使用到PIC18F4550微控制器。 427

图8-8为PIC18F4550微控制器的USB部分的总体结构图。PORTC引脚RC4（引脚23）和RC5（引脚24）被用于USB接口。RC4是USB的数据线D-引脚，而RC5是USB的数据线D+引脚。提供内部上拉电阻，如果需要，也可以禁止使用内部上拉电阻（设置UPUEN=0），并使用外部上拉电阻代替。对于全速操作，可以使用内部或外部上拉电阻连接到数据引脚D+；而对于低速操作，则应该将内部或外部上拉电阻连接到数据引脚D−。

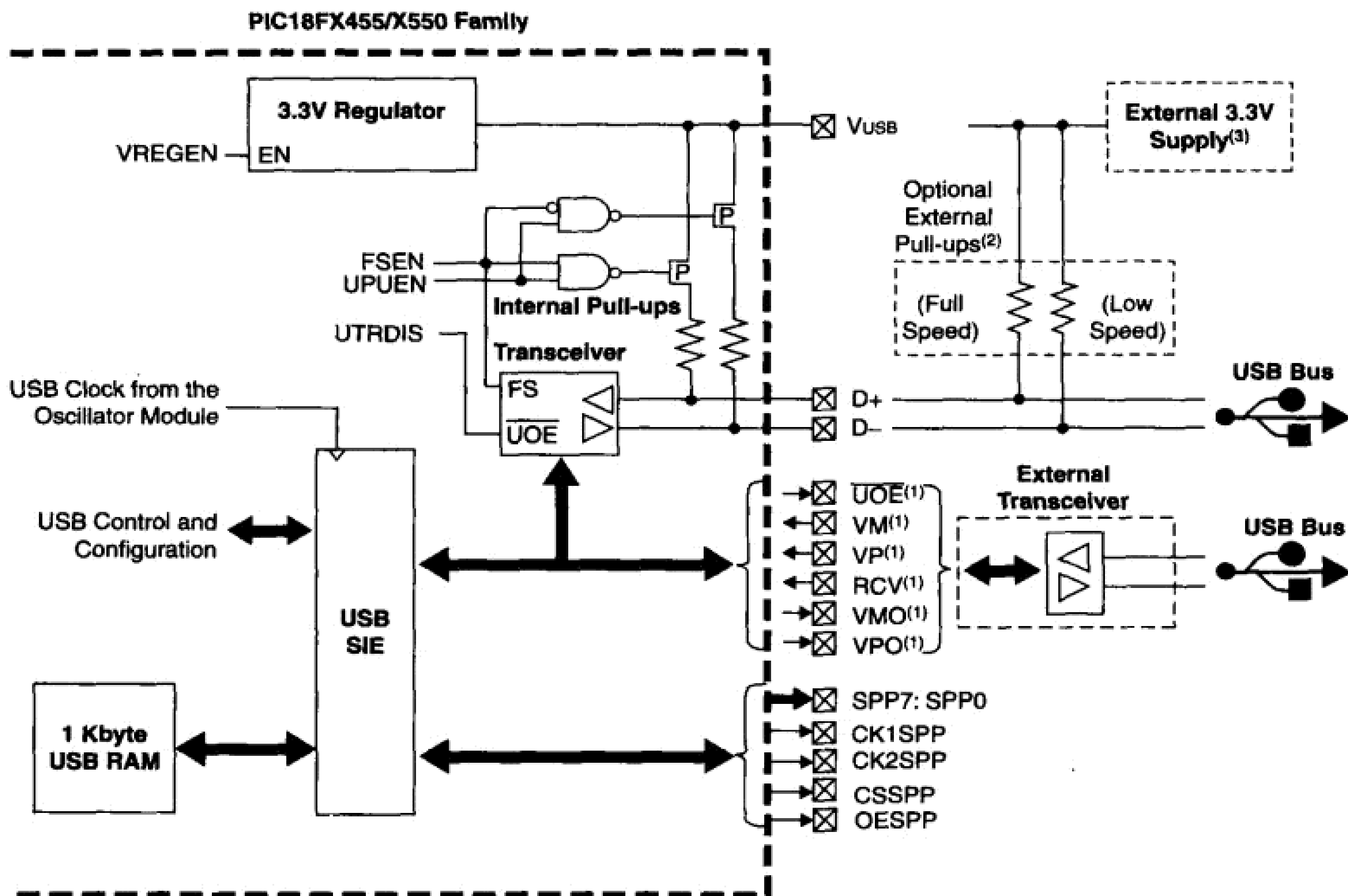
USB模块的操作配置可由3个控制寄存器来完成，总共有22个寄存器用来管理实际的USB通信。这些寄存器的配置是一件很复杂的工作，本书将不作讨论。有兴趣的读者可以参阅PIC18F4550数据表和关于USB内部结构的书籍。本章将使用mikroC语言的USB库函数来实现USB通信。这些库函数的技术细节将在8.6节讨论。 428

8.6 mikroC 语言的 USB 总线库函数

mikroC语言支持大量的函数，用于USB HID类型通信。每一个基于USB库函数的项目都应该包含一个描述符源文件，该文件包括供应商ID和名字、产品ID和名字、报告长度和其他相关信息。可以使用mikroC的集成USB HID终端工具（请参阅Tools→HID 终端）来创建描述符源文件。描述

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

符文件的默认名字为USBdsc.c，可以根据需要修改名字。USBdsc.c文件必须被包括在基于USB的项目中，这可以通过使用mikroC IDE工具或者作为项目源文件的一个#include选项来实现。



注解　1：当内部收发器被禁止时（UTRDIS=1），这个信号才有效。
2：如果使用外部上拉电阻，那么应该禁止内部上拉电阻（设置UPUEN=0）。
3：当使用3.3V的外接电源时，不要启动内部的电压调节器。

图8-8　PIC18F4550微处理器的USB结构图

当使用带有内置USB的PIC微控制器（如PIC18F4550），且端口引脚RC4和RC5分别与USB连接器的D+和D−引脚相连时，mikroC语言支持下列的USB总线库函数。

Hid_Enable。这个函数用来使能USB通信，它有两个参数：读缓冲器地址和写缓冲器地址。在调用其他USB库函数之前，应该首先调用该函数。该函数没有返回值。

Hid_Read。这个函数用来从USB总线接收数据，然后将其存储在接收缓冲器。该函数没有参数，返回的是接收到的字符数。

Hid_Write。这个函数用来将写缓冲器中的数据发送到USB总线。缓冲器的名字（初始化时也使用同样的缓冲器）和要发送的数据长度作为参数传递给函数。该函数不返回任何数据。

Hid_Disable。这个函数用来禁止USB数据传输。该函数没有参数，也不返回任何数据。

图8-9所示为PIC18F4550微控制器的USB接口。如图8-9所示，该电路接口非常简单。除了电源和接地引脚外，只需要将两个引脚连接到USB连接器。微控制器从USB端口获得工作电源。

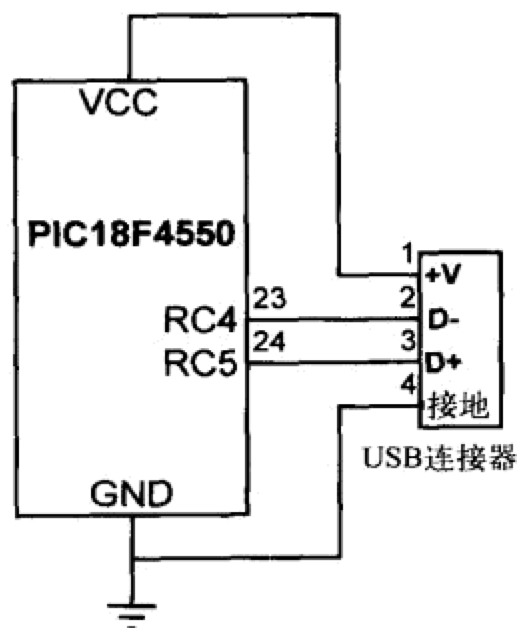


图8-9　PIC18F4550 的USB接口

429

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

项目 8.1 基于 USB 的微控制器输出端口

这个项目描述了基于USB的微控制器输出端口的设计。PIC18F4550微控制器通过一条USB线连接到PC。在PC上运行Visual Basic程序，通过USB总线向微控制器发送命令，要求微控制器设置/复位PORTB端口的I/O位。

该项目的方框图如图8-10所示。其电路图如图8-11所示。PIC18F4550微控制器的USB线连接到USB连接器。微控制器由USB线供电（不需外部电源供电）。这使得基于USB产品的设计相对便宜，在总功耗低于100 mA的应用中将很有吸引力。微控制器工作在8MHz晶振频率下。 430

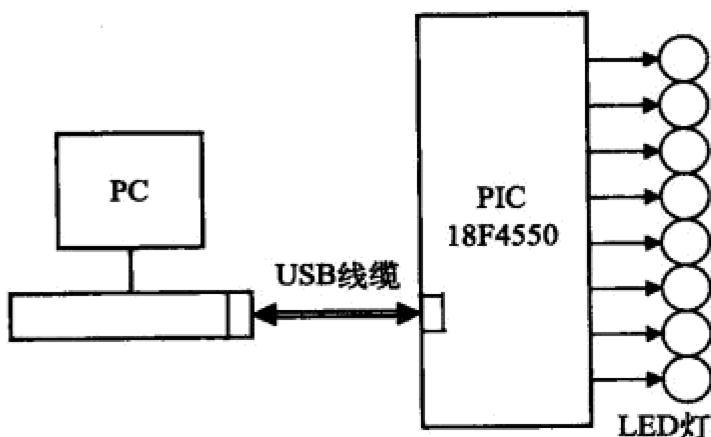


图8-10 项目的方框图

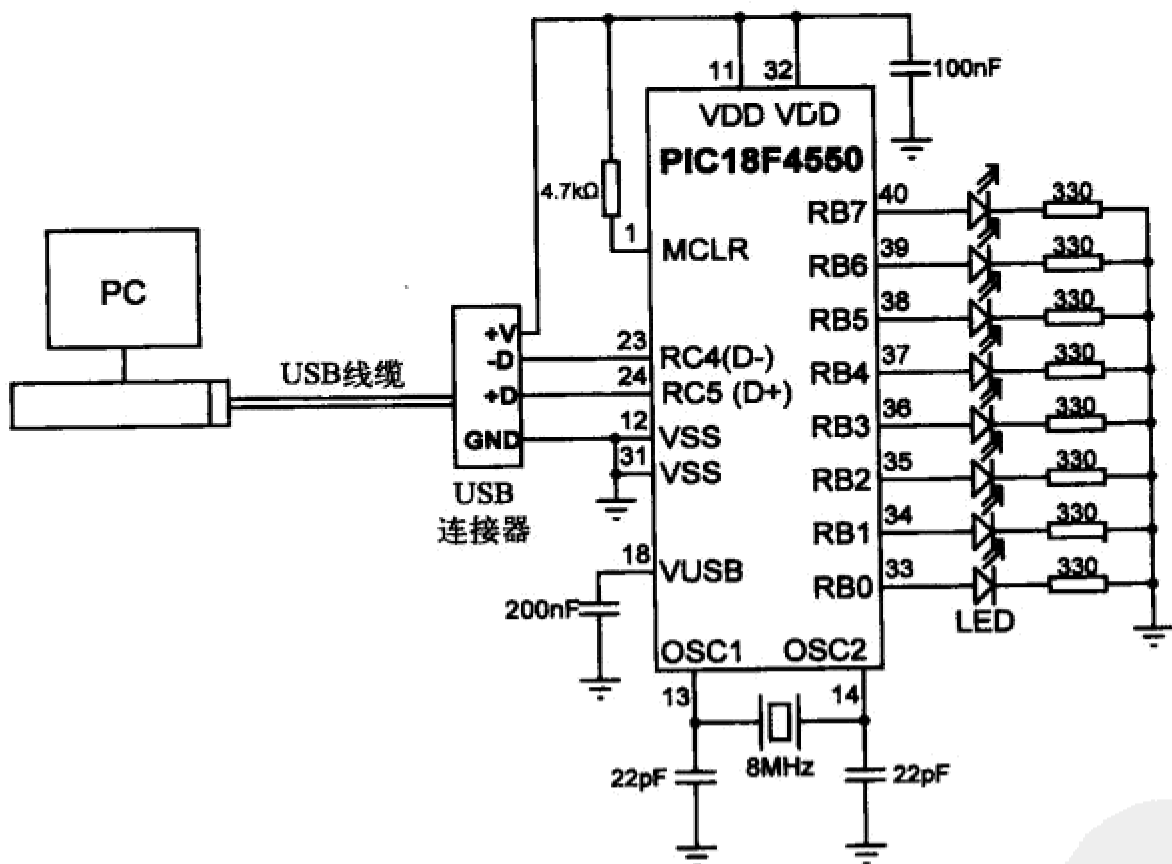


图8-11 项目的电路图

微控制器的PORTB引脚连接到LED灯上，当有命令从PC上发送过来时，LED灯会改变状态。这使得测试项目变得非常容易。注意到，为了提高稳定性，在微控制器的Vusb引脚（引脚18）和地之间应该连接有一个电容（约为200 nF）。

项目的软件由两部分组成：PC软件和微控制器软件。下面将分别介绍这两类软件。

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

PC软件

PC软件是用Visual Basic编写的。假定用户具有Visual Basic编程语言的基本知识。Visual Basic编程指令不在本书的讨论范围之内，有兴趣的读者请参阅这方面的书籍。

431

源程序列表和可执行应用程序在本书附属资源中可以找到。不想编程的读者可以使用或者修改上面已有的程序。

本例中的Visual Basic程序只包含一种界面形式，如图8-12所示。在文本框中，要求以十进制输入PORTB数据，然后用鼠标点击命令按钮“CLICK TO SEND”。例如，输入十进制数15，将会点亮连接到PORTB端口引脚RB0、RB1、RB2和RB3的LED灯。

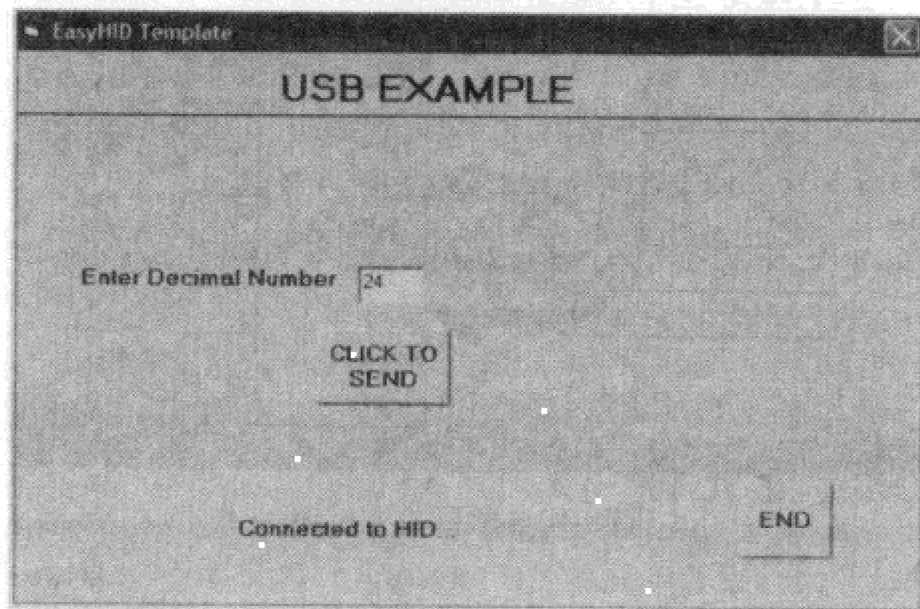


图8-12　PC的VISUAL BASIC窗口

程序将输入的数据以信息包的形式发送给微控制器，信息包由四个字符组成，格式如下：

$$P = nT$$

其中，字符P表示数据开始，字符n是指要发送到PORTB的字节数，T是结束字符。

例如，若PORTB的第3位和第4位被设置为1，也就是PORTB= “00011000”，则Visual Basic程序就会通过USB链接向微控制器发送信息包P=24T（数24是以单个二进制字节而不是以两个ASCII字节的形式发送）。这种界面形式的底部显示有连接状态。

432

本节中用到的VISUAL BASIC程序是基于USB实用程序的。该USB实用程序是Mecanique公司开发的EasyHID USB Wizard，可以从该公司的网站（www.mecanique.co.uk）上免费下载。EasyHID是专为USB2.0设计的，并不需要研发驱动程序，这犹如XP操作系统已带有一个基于HID的USB驱动程序一样。该实用程序为使用HID类型设备接口的USB应用的PC终端生成VISUAL BASIC、C++或Borland Delphi模板代码。除此之外，utility还能为PIC18F4550和其他类似的微控制器生成基于Proton Development Suite（www.crownhill.co.uk）、Swordish PIC Basic或者PicBasic Pro（www.melabs.com）编程语言的USB模板代码。这种生成的模板代码可与用户代码一起进行扩展，以实现所期望的应用。

生成VISUAL BASIC代码模板的步骤如下。

- 单击“Download EasyHID as a Standalone Application”，从Mecanique公司网站上下载EasyHID压缩文件。
- 解压文件，双击SETUP安装程序。
- 在程序开始时，将看到如图8-13所示的窗口界面。在公司名称，产品名称和序列号栏输入数据。
- 输入供应商ID（VID）和产品ID（PID），如图8-14所示。供应商ID在全球范围内是独一无二的，

433

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

由USB组织（www.usb.org）给定。Mecanique公司拥有一个供应商ID并且可以低价向你颁发产品ID。有了独特的VID和PID，你的产品就可以运到世界各地销售。在该例中，出于测试目的考虑，设置VID=4660，PID=1。

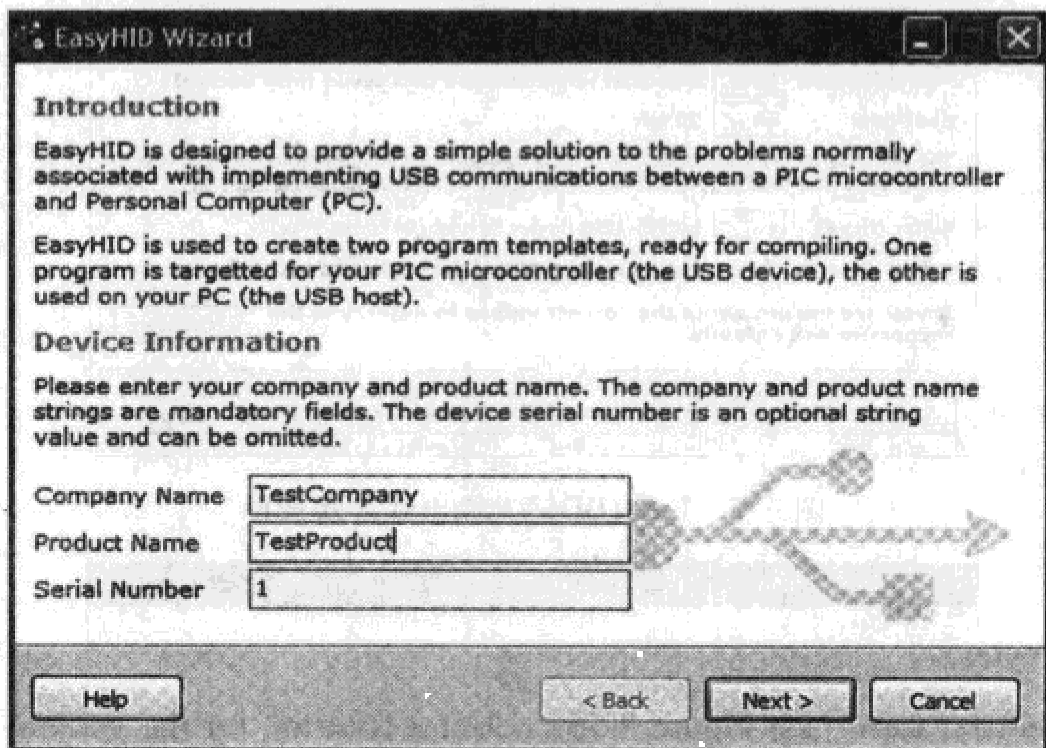


图8-13　EasyHID的第一个窗口界面

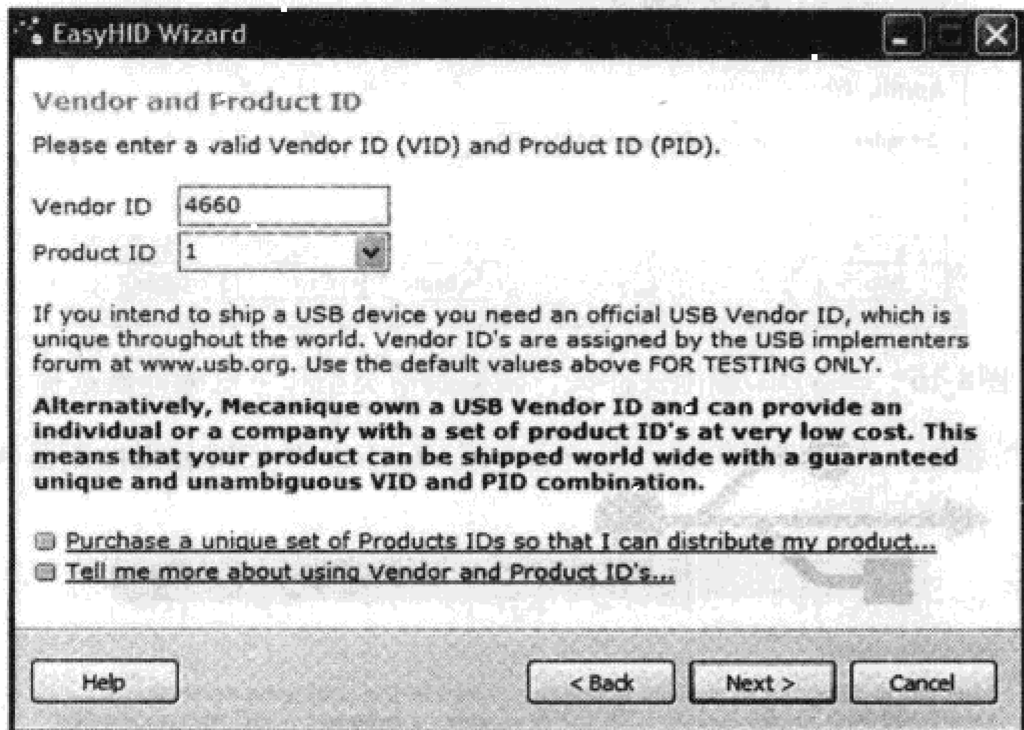


图8-14　EasyHID VID和PID输入窗口

- 单击“Next”按钮，将显示如图8-15所示的窗口。这里，重要的参数是输入输出缓冲器的大小，它们用来指明在PC和微控制器之间进行USB数据通信时，接收和发送的字节数量。在该例中，两个字段都选择为4 B（输出采用P=*n*T格式，即4 B）。
- 在下一个窗口（如图8-16所示），给要生成的文件选择一个路径，选择要使用到的微控制器编译器（这一字段并不重要，因为只要生成VISUAL BASIC代码（即PC）），选择微控制器的类型，最后选择VISUAL BASIC作为要使用的语言。

434

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

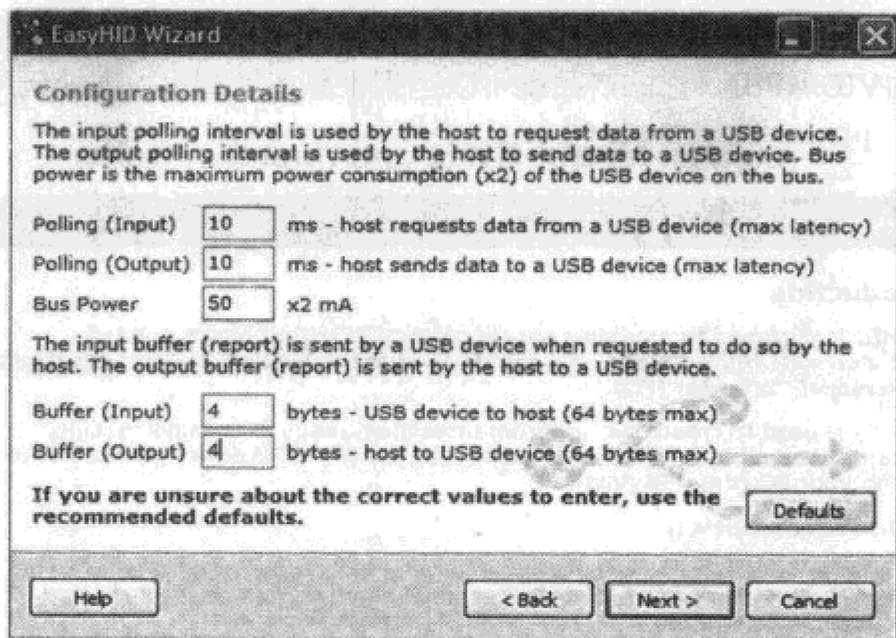


图8-15 EasyHID输入输出缓冲器选择

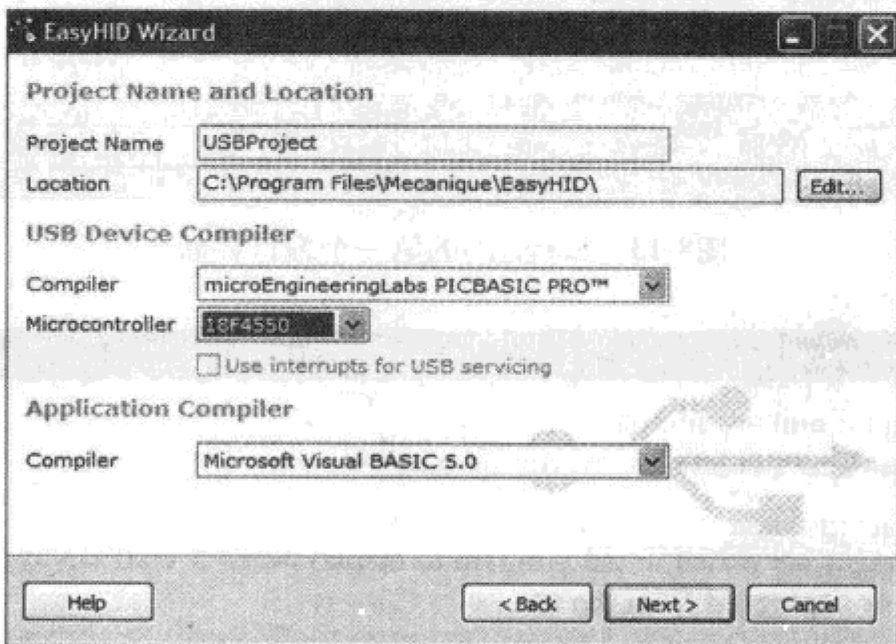


图8-16 EasyHID输出文件夹、微控制器类型和主机编译器选择

■ 单击“Next”按钮，在已选择的路径里就会生成VISUAL BASIC和微控制器代码模板（如图8-17所示的最后一个窗口界面）。

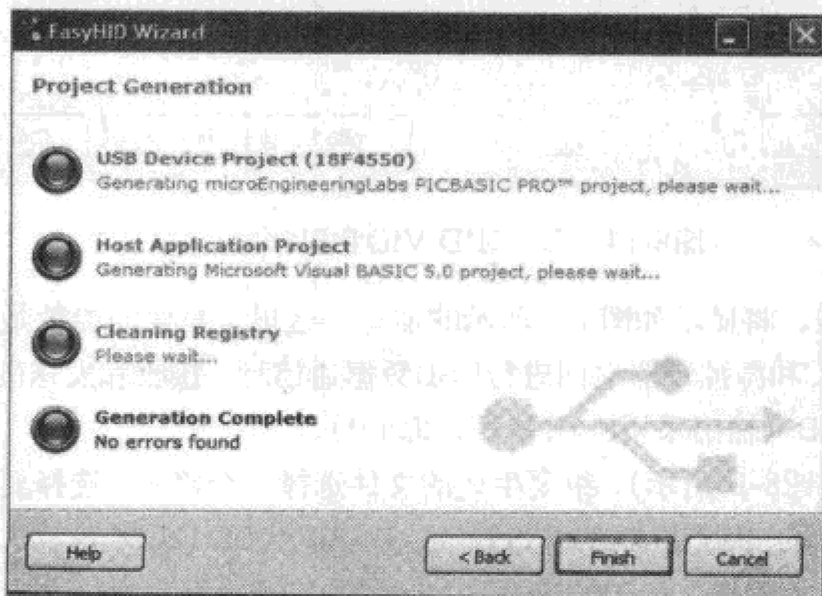


图8-17 EasyHID最后一个窗口

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

图8-18给出了使用EasyHID向导生成的VISUAL BASIC文件。文件基本上包括一个空白表（FormMain.frm）、一个模块文件（mcHIDInterface.BAS）和一个项目文件（USBProject.vbp）。

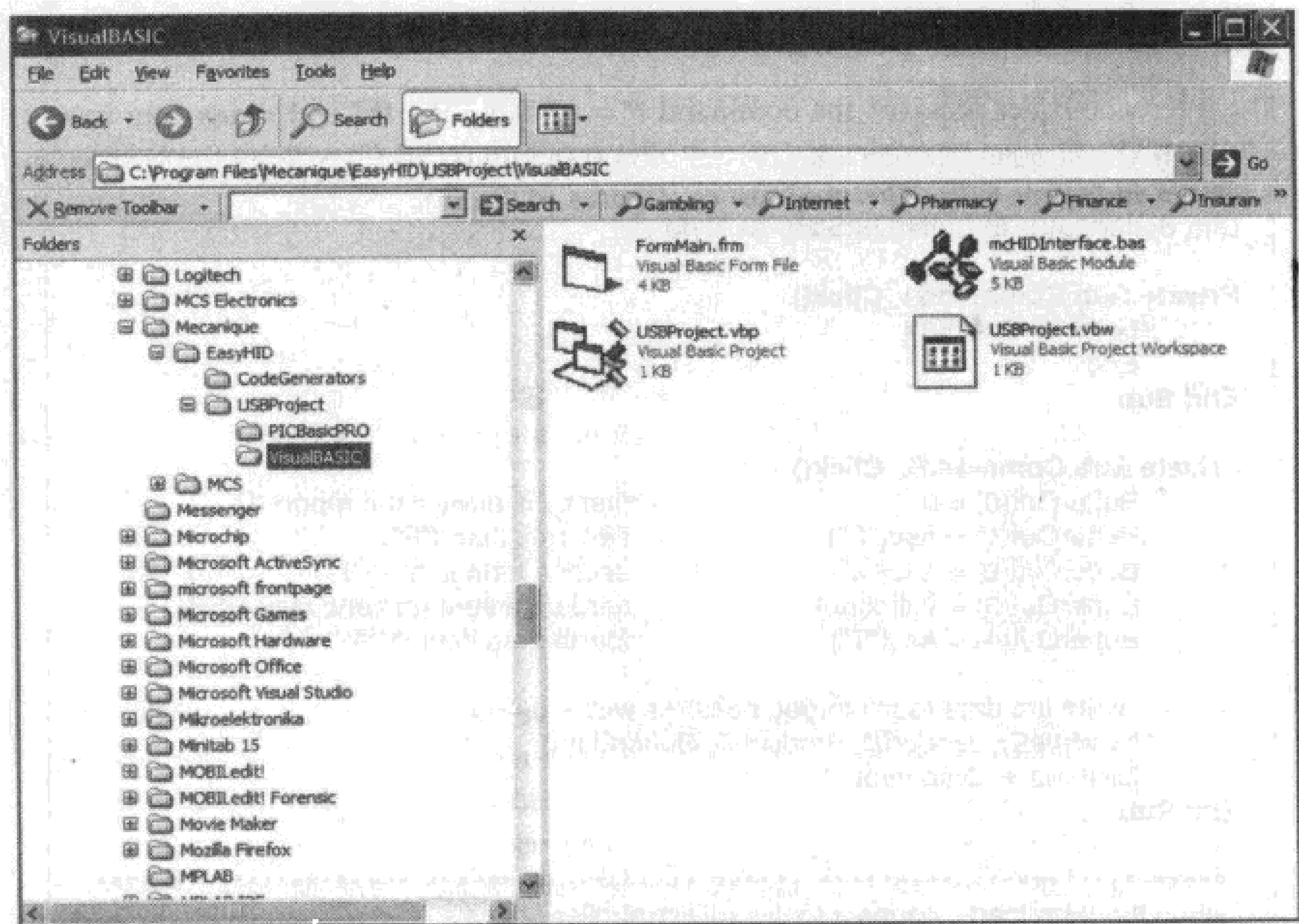


图8-18 使用EasyHID向导生成的文件

在本项目中，将使用EasyHID向导生成的文件作如下的修改。

- 修改空白表，用来显示不同的控制，如图8-12所示。
- 当有USB设备连接到PC或从PC上拔出时，相应的信息将会被添加到显示程序中。
- 添加一个子程序，在点击CLICK TO SEND按钮时，子程序可以读入用户输入的数据并通过USB总线将数据发送到微控制器。其程序代码如下：

435
~
437

```
Private Sub Command2_Click()

    BufferOut(0) = 0              ' first by is always the report ID
    BufferOut(1) = Asc("P")       ' first data item ("P")
    BufferOut(2) = Asc("=")       ' second data item ("=")
    BufferOut(3) = Val(txtno)     ' third data item (number to send)
    BufferOut(4) = Asc("T")       ' fourth data item ("T")

    ' write the data (don't forget, pass the whole array)...
    hidWriteEx VendorID, ProductID, BufferOut(0)
    lblstatus = "Data sent..."
End Sub
```

BufferOut用来存储要通过USB总线发送给微控制器的数据。注意到，缓冲器的第一个字节是报告ID，并且必须被设置为0。实际的数据则从数组的地址BufferOut(1)开始，数据以前面所述的P=*n*T格式发送。在数据发送完毕后，在显示窗的底部会出现信息“Data sent ...”。

图8-19给出了VISUAL BASIC程序的最终清单。程序分为两部分：表USB1.FRM和模块USB1.BAS。这些程序需要下载到VISUAL BASIC开发环境中使用。本书附属资源中有该程序的一个可安装版本，供没有VISUAL BASIC开发环境的用户使用。程序的安装与普通的Windows 软件的安装过程一样。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
USB1.FRM

' vendor and product IDs
Private Const VendorID = 4660
Private Const ProductID = 1

' read and write buffers
Private Const BufferInSize = 8
Private Const BufferOutSize = 8
Dim BufferIn(0 To BufferInSize) As Byte
Dim BufferOut(0 To BufferOutSize) As Byte

Private Sub Command1_Click()
        Form_Unload (0)
        End
End Sub

Private Sub Command2_Click()
        BufferOut(0) = 0                ' first by is always the report ID
        BufferOut(1) = Asc("P")         ' first data item ("P")
        BufferOut(2) = Asc("=")         ' second data item ("-")
        BufferOut(3) = Val(txtno)       ' third data item (to send over USB)
        BufferOut(4) = Asc("T")         ' fourth data item ("T")

        ' write the data (don't forget, pass the whole array)...
        hidWriteEx VendorID, ProductID, BufferOut(0)
        lblstatus = "Data sent..."
End Sub

' ***************************************************************************
' when the form loads, connect to the HID controller - pass
' the form window handle so that you can receive notification
' events...
'****************************************************************************
Private Sub Form_Load()
  ' do not remove!
        ConnectToHID (Me.hwnd)
        lblstatus = "Connected to HID..."
End Sub

'****************************************************************************
' disconnect from the HID controller...
'****************************************************************************
Private Sub Form_Unload(Cancel As Integer)
        DisconnectFromHID
End Sub


'****************************************************************************
' a HID device has been plugged in...
'****************************************************************************
Public Sub OnPlugged(ByVal pHandle As Long)
  If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle)=
ProductID Then
        lblstatus = "USB Plugged....."
        End If
End Sub

'****************************************************************************

' a HID device has been unplugged...
```

图8-19 用于USB连接的PC端的VISUAL BASIC程序

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
'*****************************************************************************
Public Sub OnUnplugged(ByVal pHandle As Long)
  If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) =
ProductID Then
        lblstatus = "USB Unplugged...."
  End If
End Sub

'*****************************************************************************

' controller changed notification - called
' after ALL HID devices are plugged or unplugged
'*****************************************************************************
Public Sub OnChanged()
  Dim DeviceHandle As Long

        ' get the handle of the device we are interested in, then set
        ' its read notify flag to true - this ensures you get a read
        ' notification message when there is some data to read...
        DeviceHandle = hidGetHandle(VendorID, ProductID)
        hidSetReadNotify DeviceHandle, True
End Sub

'*****************************************************************************

' on read event...
'*****************************************************************************
Public Sub OnRead(ByVal pHandle As Long)

  ' read the data (don't forget, pass the whole array)...
  If hidRead(pHandle, BufferIn(0)) Then
    ' ** YOUR CODE HERE **
    ' first byte is the report ID, e.g. BufferIn(0)
    ' the other bytes are the data from the microcontrolller...
  End If
End Sub
```

USB1.BAS

```
' this is the interface to the HID controller DLL - you should not
' normally need to change anything in this file.
'
' WinProc() calls your main form 'event' procedures - these are currently
' set to..
'
' MainForm.OnPlugged(ByVal pHandle as long)
' MainForm.OnUnplugged(ByVal pHandle as long)
' MainForm.OnChanged()
' MainForm.OnRead(ByVal pHandle as long)

Option Explicit

' HID interface API declarations...
Declare Function hidConnect Lib "mcHID.dll" Alias "Connect" (ByVal pHostWin As
Long) As Boolean
Declare Function hidDisconnect Lib "mcHID.dll" Alias "Disconnect" () As Boolean
Declare Function hidGetItem Lib "mcHID.dll" Alias "GetItem" (ByVal pIndex As
Long) As Long
Declare Function hidGetItemCount Lib "mcHID.dll" Alias "GetItemCount" () As
Long
```

图8-19 （续）

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源


```
Declare Function hidRead Lib "mcHID.dll" Alias "Read" (ByVal pHandle As Long,
ByRef pData As Byte) As Boolean
Declare Function hidWrite Lib "mcHID.dll" Alias "Write" (ByVal pHandle As Long,
ByRef pData As Byte) As Boolean
Declare Function hidReadEx Lib "mcHID.dll" Alias "ReadEx" (ByVal pVendorID As
Long, ByVal pProductID As Long, ByRef pData As Byte) As Boolean
Declare Function hidWriteEx Lib "mcHID.dll" Alias "WriteEx" (ByVal pVendorID
As Long, ByVal pProductID As Long, ByRef pData As Byte) As Boolean
Declare Function hidGetHandle Lib "mcHID.dll" Alias "GetHandle" (ByVal
pVendoID As Long, ByVal pProductID As Long) As Long
Declare Function hidGetVendorID Lib "mcHID.dll" Alias "GetVendorID" (ByVal
pHandle As Long) As Long
Declare Function hidGetProductID Lib "mcHID.dll" Alias "GetProductID" (ByVal
pHandle As Long) As Long
Declare Function hidGetVersion Lib "mcHID.dll" Alias "GetVersion" (ByVal
pHandle As Long) As Long
Declare Function hidGetVendorName Lib "mcHID.dll" Alias "GetVendorName"
(ByVal pHandle As Long, ByVal pText As String, ByVal pLen As Long) As Long
Declare Function hidGetProductName Lib "mcHID.dll" Alias "GetProductName"
(ByVal pHandle As Long, ByVal pText As String, ByVal pLen As Long) As Long
Declare Function hidGetSerialNumber Lib "mcHID.dll" Alias"GetSerialNumber"
(ByVal pHandle As Long, ByVal pText As String, ByVal pLen As Long) As Long
Declare Function hidGetInputReportLength Lib "mcHID.dll" Alias
"GetInputReportLength" (ByVal pHandle As Long) As Long
Declare Function hidGetOutputReportLength Lib "mcHID.dll" Alias
"GetOutputReportLength" (ByVal pHandle As Long) As Long
Declare Sub hidSetReadNotify Lib "mcHID.dll" Alias "SetReadNotify" (ByVal
pHandle As Long, ByVal pValue As Boolean)
Declare Function hidIsReadNotifyEnabled Lib "mcHID.dll" Alias
"IsReadNotifyEnabled" (ByVal pHandle As Long) As Boolean
Declare Function hidIsAvailable Lib "mcHID.dll" Alias "IsAvailable" (ByVal
pVendorID As Long, ByVal pProductID As Long) As Boolean

' windows API declarations - used to set up messaging...
Private Declare Function CallWindowProc Lib "user32" Alias "CallWindowProcA"
(ByVal lpPrevWndFunc As Long, ByVal hwnd As Long, ByVal Msg As Long,
ByVal wParam As Long, ByVal lParam As Long) As Long
Private Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA"
(ByVal hwnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As
Long

' windows API Constants
Private Const WM_APP = 32768
Private Const GWL_WNDPROC = -4

' HID message constants
Private Const WM_HID_EVENT = WM_APP + 200
Private Const NOTIFY_PLUGGED = 1
Private Const NOTIFY_UNPLUGGED = 2
Private Const NOTIFY_CHANGED = 3
Private Const NOTIFY_READ = 4

' local variables
Private FPrevWinProc As Long   ' Handle to previous window procedure
Private FWinHandle As Long    ' Handle to message window

' Set up a windows hook to receive notification
' messages from the HID controller DLL - then connect
' to the controller
Public Function ConnectToHID(ByVal pHostWin As Long) As Boolean
```

图8-19 （续）

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com


```
   FWinHandle = pHostWin
   ConnectToHID = hidConnect(FWinHandle)
   FPrevWinProc = SetWindowLong(FWinHandle, GWL_WNDPROC, AddressOf
WinProc)
End Function

' Unhook from the HID controller and disconnect...
Public Function DisconnectFromHID() As Boolean
   DisconnectFromHID = hidDisconnect
   SetWindowLong FWinHandle, GWL_WNDPROC, FPrevWinProc
End Function

' This is the procedure that intercepts the HID controller messages...
Private Function WinProc(ByVal pHWnd As Long, ByVal pMsg As Long,
ByVal wParam As Long, ByVal lParam As Long) As Long
   If pMsg = WM_HID_EVENT Then
      Select Case wParam

       ' HID device has been plugged message...
       Case Is = NOTIFY_PLUGGED
          MainForm.OnPlugged (lParam)

       ' HID device has been unplugged
       Case Is = NOTIFY_UNPLUGGED
          MainForm.OnUnplugged (lParam)

       ' controller has changed...
       Case Is = NOTIFY_CHANGED
          MainForm.OnChanged

       ' read event...
       Case Is = NOTIFY_READ
          MainForm.OnRead (lParam)
       End Select

   End If

   ' next...
   WinProc = CallWindowProc(FPrevWinProc, pHWnd, pMsg, wParam, lParam)

End Function
```

图8-19 （续）

微控制器软件

微控制器从PC接收命令P=*n*T，并向PORTB发送字节数*n*。图8-20给出了不包括USB代码的微控制器程序（USB.C）清单。程序将PORTB配置为数字输出端口。

```
/**************************************************************************
          USB BASED MICROCONTROLLER OUTPUT PORT
          =========================================

In this project a PIC18F4550 type microcontroller is connected to a PC through
the USB link.

A Visual Basic program runs on the PC where the user enters the bits to be set
or cleared on PORTB of the microcontroller. The PC sends a command to the
microcontroller requesting it to set or reset the required bits of the microcontroller
PORTB.
```

图8-20 不包括USB代码的微控制器程序

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
The command sent by the PC to the microcontroller is in the following format:

        P=nT

where n is the byte the microcontroller is requested to send to PORTB of the
microcontroller.

Author:      Dogan Ibrahim
Date:        September 2007
File:        USB.C
*****************************************************************************/

void main()
{
    ADCON1 = 0xFF;                       // Set PORTB to digital I/O
    TRISB = 0;                           // Set PORTB to outputs
    PORTB = 0;                           // Clear all outputs
}
```

图8-20 （续）

生成USB描述符文件

在mikroC程序的开始，必须包括USB描述符文件。使用mikroC编译器的Tools菜单创建描述符文件的步骤如下。

- 选择Tools→HID Terminal。
- 弹出新的窗口。点击描述符标签，弹出如图8-21所示的窗口。

图8-21 创建USBdsc描述符文件

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

- 在这里要输入的重要参数有供应商ID、产品ID、输入缓冲器大小、输出缓冲器大小、供应商名字（VN）和产品名（PN）。注意，VID和PID是十六进制格式的，当要使用EasyHID向导生成代码时，以上参数必须跟VISUAL BASIC程序的参数保持一致。选择VID=1234（等效于十进制数6460）、PID=1、输入缓冲器大小为4、输出缓冲器大小为4、VN和PN为任意名字。
- 查看mikroC编译器。
- 单击"CREATE"按钮，将会要求填写文件名，然后在这个文件夹里创建描述符文件USBdsc。将该文件带上".C"扩展名（即文件的全名为USBdsc.C），然后将这个文件赋值到下面的文件夹里（由于所需的其他mikroC文件已经在这个文件夹中，所以在这里复制USBdsc.C文件也是合理的）。

```
C:\Program Files\Mikroelektronika\mikroC\Examples\EasyPic4
\extra_examples\HID-library\USBdsc.c
```

438～442

千万不要修改USBdsc.C文件的内容。本书附属资源中附有这个文件的程序清单。

图8-22给出了带有USB代码的微控制器程序（程序USB1.C）清单。在程序的开始处，将USB描述符文件USBdsc.C包含进来。USB连接操作要求微控制器每隔几毫秒就发送一个激活信号给PC，以保证微控制器与PC之间保持连接。这可以使用TIMER0设置一个定时器中断服务子程序来实现。在定时器中断子程序内，调用mikroC USB 函数`HID_InterruptProc`。在从中断服务子程序返回之前，重装定时器TMRL初值，重新使能定时器中断。

443

```
/*****************************************************************************
            USB BASED MICROCONTROLLER OUTPUT PORT
            ======================================

In this project a PIC18F4550 type microcontroller is connected
to a PC through the USB link.

A Visual Basic program runs on the PC where the user enters the bits to be set or
cleared on PORTB of the microcontroller. The PC sends a command to the
microcontroller requesting it to set or reset the required bits of the microcontroller
PORTB.

A 8MHz crystal is used to operate the microcontroller. The actual CPU clock is raised
to 48MHz by setting configuration bits. Also, the USB module is operated with
48MHz.

The command sent by the PC to the microcontroller is in the following format:

      P=nT

where n is the byte the microcontroller is requested to send to PORTB of the
microcontroller.

This program includes the USB code.

Author:      Dogan Ibrahim
Date:        September 2007
File:        USB1.C
*****************************************************************************/

#include "C:\Program
Files\Mikroelektronika\mikroC\Examples\EasyPic4\extra_examples\HID-
library\USBdsc.c"

unsigned char Read_buffer[64];
```

图8-22 包括USB代码的微控制器程序

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
unsigned char Write_buffer[64];
unsigned char num;
//
// Timer interrupt service routine
//
void interrupt()
{
  HID_InterruptProc();                          // Keep alive
  TMR0L = 100;                                  // Re-load TMR0L
  INTCON.TMR0IF = 0;                            // Re-enable TMR0 interrupts
}


//
// Start of MAIN program
//
void main()
{

    ADCON1 = 0xFF;                        // Set PORTB to digital I/O
    TRISB = 0;                            // Set PORTB to outputs
    PORTB = 0;                            // Clear all outputs
//
// Set interrupt registers to power-on defaults
// Disable all interrupts
//
    INTCON=0;
    INTCON2=0xF5;
    INTCON3=0xC0;
    RCON.IPEN=0;
    PIE1=0;
    PIE2=0;
    PIR1=0;
    PIR2=0;
//
// Configure TIMER 0 for 3.3ms interrupts. Set prescaler to 256
// and load TMR0L to 100 so that the time interval for timer
// interrupts at 48MHz is 256*(256-100)*0.083 = 3.3ms
//
// The timer is in 8-bit mode by default
//
  T0CON  = 0x47;                          // Prescaler = 256
  TMR0L  = 100;                           // Timer count is 256-156 = 100
  INTCON.TMR0IE = 1;                      // Enable T0IE
  T0CON.TMR0ON = 1;                       // Turn Timer 0 ON
  INTCON = 0xE0;                          // Enable interrupts

//
// Enable USB port
//
  Hid_Enable(&Read_buffer, &Write_buffer);
  Delay_ms(1000);
  Delay_ms(1000);
//
// Read from the USB port. Number of bytes read is in num
//

  for(;;)                           // do forever
{
    num=0;
    while(num != 4)                 // Get 4 characters
```

图8-22 （续）

PDG

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

tyw藏书



图8-22 （续）

在主程序里，PORTB被定义为数字I/O口。当TRISB被清零时，所有PORTB引脚都被设置为输出引脚。为安全起见，所有中断寄存器都被设置为上电复位值。然后，设置定时器中断。定时器工作在8位模式，使用预分频器值256。虽然晶体振荡器时钟频率为8 MHz，但CPU的运行时钟频率为48 MHz，正如稍后所述。选择定时器初值TMR0L=100，时钟频率为48 MHz（CPU的时钟周期为0.083 μs），于是定时器中断的间隔时间为：

$$(256-100)\times 256\times 0.083\ \mu s$$

即约为3.3 ms。因此，每隔3.3 ms就需要发送一个激活消息。

444

然后，调用函数Hid_Enable使能USB端口。此时，程序进入一个无限循环，并使用函数Hid_Read 从USB端口读取数据。当接收到的4 B的格式全部正确（即字节0＝“P”，字节1＝“＝”，字节3＝“T”）时，从字节2读取数据字节，并发送到微控制器的PORTB。

值得注意的是，当使用函数Hid_Read读取数据完成时，函数返回所接收到的字节数。另外，接收到的第一个字节是第一个实际数据字节，而不是报告ID。

微控制器时钟

PIC18F4550微控制器的USB模块需要48 MHz的时钟。另外，微控制器CPU需要的时钟范围为0～48 MHz。在这个项目中，将CPU时钟设置为48 MHz。

445

可以用多种方式来提供要求的时钟脉冲。

图8-23所示为PIC18F4550的部分时钟电路。电路包括一个1:1～1:12的PLL预分频器复用器、一个4:96MHz的PLL 、一个1:2～1:6的PLL后分频器和一个1:1～1:4振荡器后分频器。假定晶振频率为8 MHz，而微控制器需要运行在48 MHz的时钟频率，并且USB模块也需要48 MHz的时钟频率，因此在mikroC IDE的编辑项目（Edit Project）选项中可作如下的设置。

446～448

- 设置_PLL_DIV2_1L，将8 MHz的时钟频率除以2，在PLL预分频器复用器的输出端产生4 MHz的时钟频率。现在，4:96MHz的PLL的输出频率为96 MHz。继续将时钟频率除以2，可以在多路复用器USBDIV的输入端产生48 MHz的时钟信号。
- 选中_USBDIV_2_1L，用来向USB模块提供48 MHz的时钟频率，并且选择除以2，用于PLL后分频器。
- 选中_CPUDIV_OSC1_PLL2_1L，用来选择PLL为时钟源。
- 选中_FOSC_HSPLL_HS_1L，用来为CPU选择48 MHz的时钟频率。
- 在mikroC IDE里设置CPU时钟频率为48 MHz（使用编辑项目）。

图8-24给出了在48 MHz的CPU时钟频率下，用于48 MHz的USB操作所选择的时钟位。

除了设置时钟位之外，还建议设置其他的配置位。下面的清单列出了在IDE 的编辑项目选项中应该设置的所有位（大部分设置都是上电复位值）：

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源

tyw藏书

```
PLLDIV_2_1L
CPUDIV_OSC1_PLL2_1L
USBDIV_2_1L

FOSC_HSPLL_HS_1H
FCMEM_OFF_1H
IESO_OFF_1H

PWRT_ON_2L
BOR_ON_2L
BORV_43_2L
VREGEN_ON_2L

WDT_OFF_2H
WDTPS_256_2H

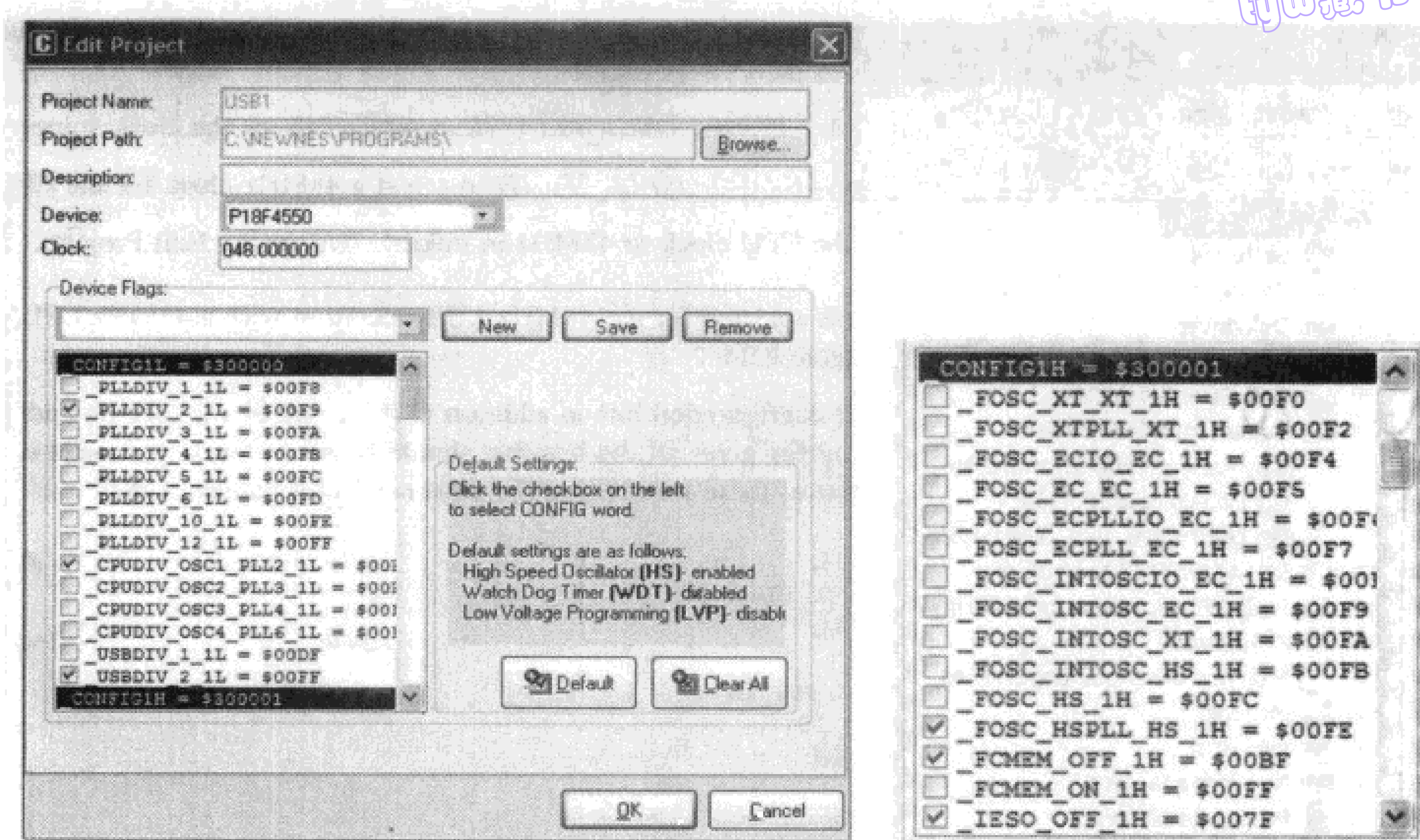
MCLRE_ON_3H
LPT1OSC_OFF_3H
PBADEN_OFF_3H
CCP2MX_ON_3H

STVREN_ON_4L
LVP_OFF_4L
ICPRT_OFF_4L
XINST_OFF_4L
DEBUG_OFF_4L
```

449

图8-23　PIC18F4550微控制器时钟电路

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源



450

图8-24　为USB操作选择时钟位

测试项目

测试项目相对容易一些，步骤如下：

- 构建硬件
- 下载程序（如图8-22所示）到PIC18F4550微控制器
- 复制或运行基于PC的VISUAL BASIC程序

当微控制器连接到PC的USB端口时，在电脑屏幕的右下方可以看到一条信息，如图8-25所示。该信息表明新的USB HID设备已经接入并被电脑识别。

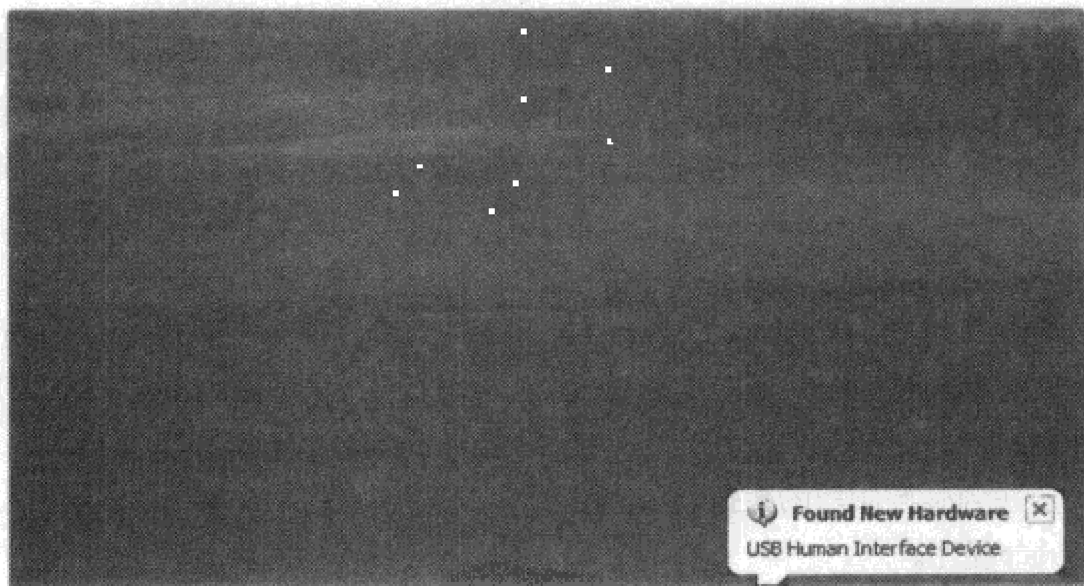


图8-25　USB连接信息

除此之外，设备管理器窗口也会显示一个HID兼容设备和一个USB HID设备，如图8-26所示。这些驱动器的特性表明，该设备的VIP为0x:1234，PID为1。

在VISUAL BASIC窗口中输入数据，单击CLICK TO SEND按钮。相应的微控制器LED灯将会被点亮。例如，若输入3，则LED灯 0和1将被点亮。

451

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

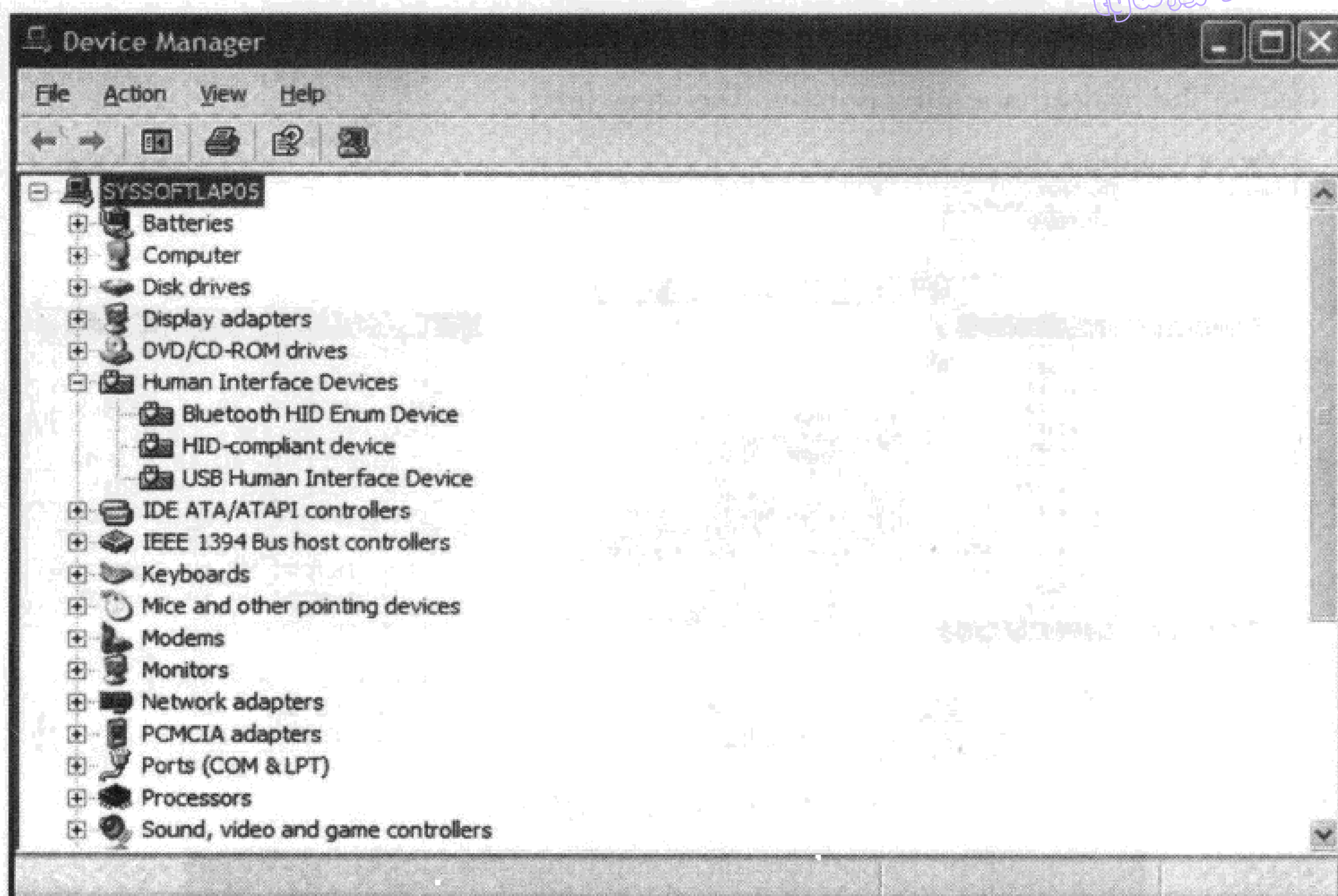


图8-26　设备管理器显示USB设备

使用USB协议分析器

不管是什么原因，只要项目不工作，都可以使用USB协议分析器来查看USB总线上的数据通信情况。市场上有很多USB协议分析器。一些昂贵的专业的分析器都是基于硬件的，需要购置特别的硬件才能使用。大部分低廉的USB协议分析器都是基于软件的。在此，简要地介绍两个基于软件的工具。

UVCView

452

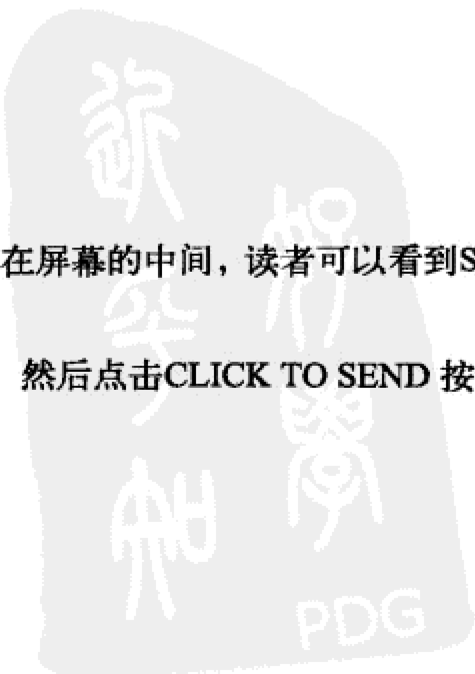
UVCView是运行在PC上的免费的微软产品，当有USB设备接入后，将显示出该设备的描述符。如图8-27所示，在微控制器被接入到PC后，UVCView将它显示出来。显示窗口的左边为系统中可用的USB端口。点击其中的一个设备，在屏幕的中间将会显示该设备描述符的具体信息。图8-27给出了设备的描述符。当要查看设备描述符的很多项时，UVCView显示是很有用的。

USBTrace

453

USBTrace是SysNucleus（www.sysnucleus.com）开发的USB协议分析器软件，可运行在PC上的。该软件监控PC的USB端口并显示总线上的所有通信情况。当要监控并记录总线上的所有通信时，这个工具就非常有用了。在制造商的网址上可以下载到一个有使用时限的USBTrace演示版。下面是一个使用程序的例子，用来观察从PC发送到微控制器的数据。

- 打开USBTrace程序。
- 将微控制器连接到PC的USB端口。
- 选中恰当的方框，从显示窗口的左边选择设备。
- 打开VISUAL BASIC程序。
- 在USBTrace菜单的左上方点击绿色的箭头，开始捕捉数据。在屏幕的中间，读者可以看到START OF LOG信息。
- 在VISUAL BASIC窗口中输入3，点亮PORTB的LED灯0和1，然后点击CLICK TO SEND 按钮。



世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

■ 可以在屏幕的中间看到数据包，如图8-28所示。

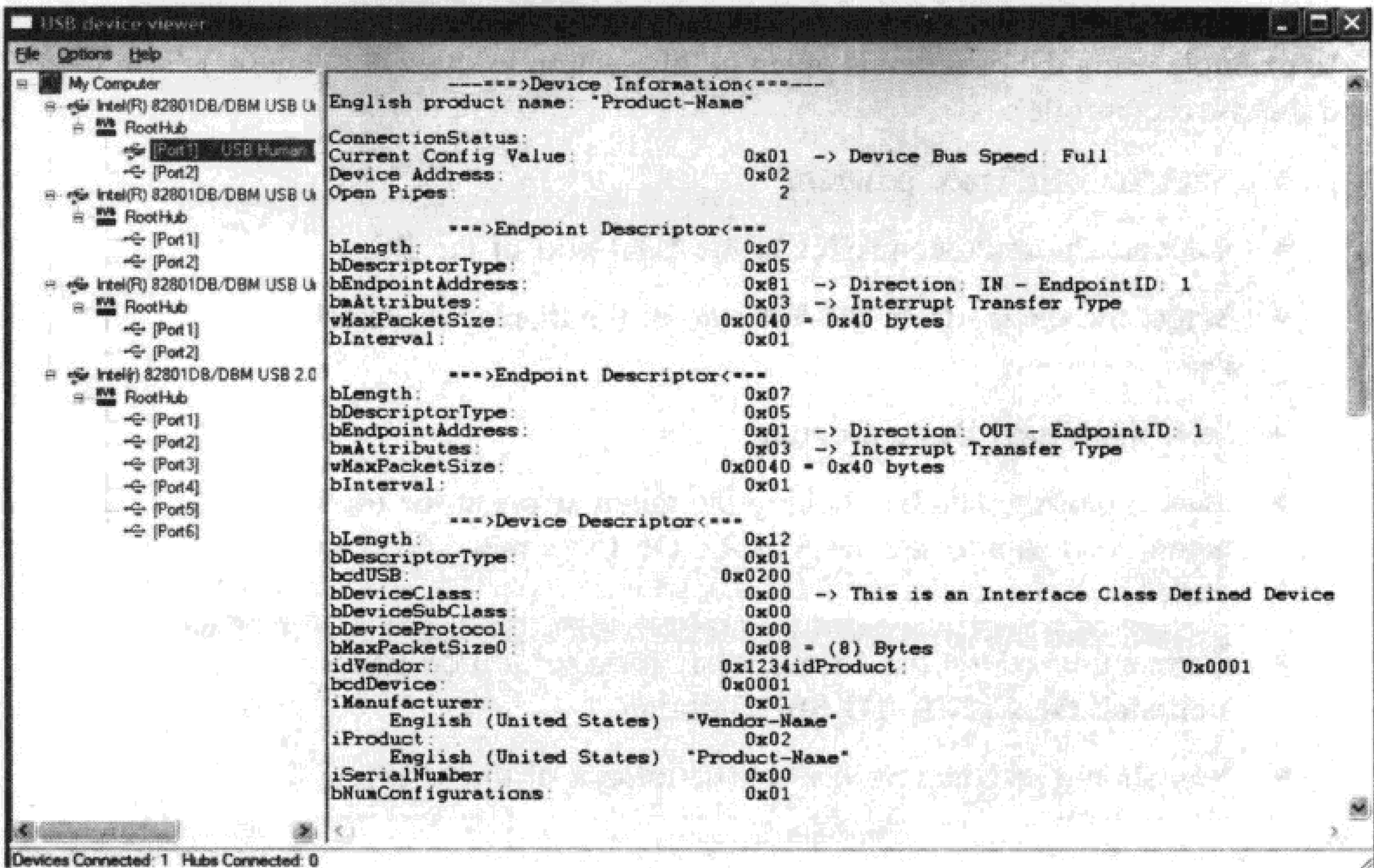


图8-27 项目的UVCView显示

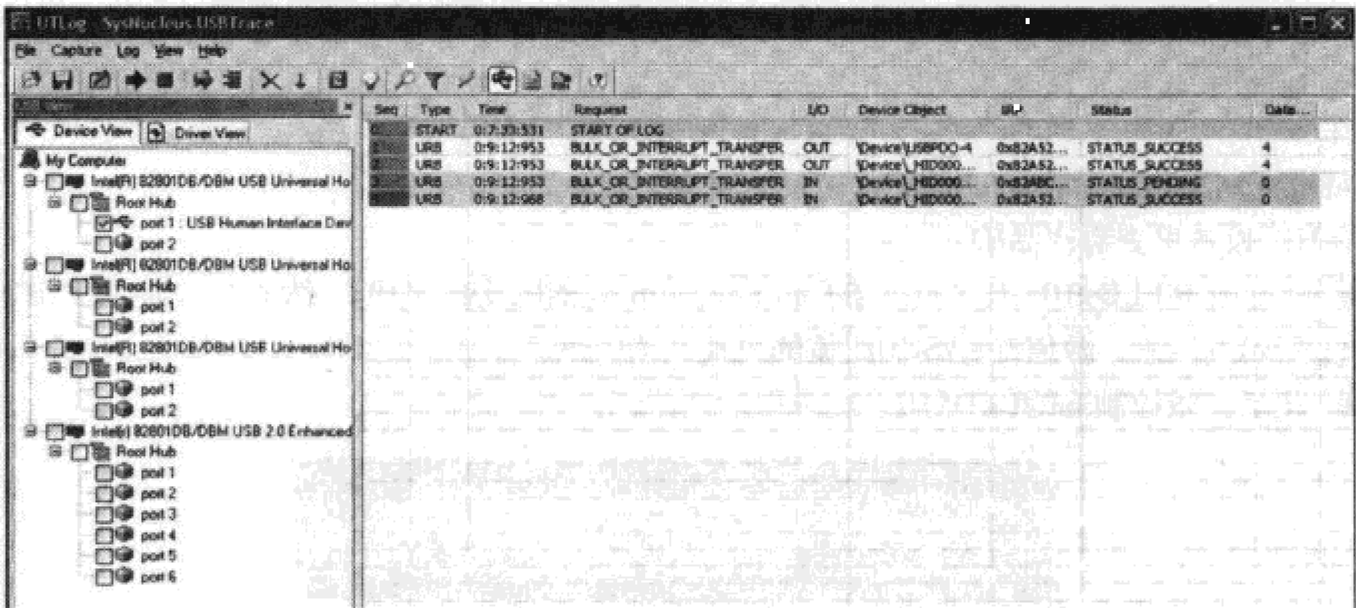


图8-28 单击CLICK TO SEND时的总线通信

■ 移动光标到第一个数据包。这是从PC发送到微控制器的数据包（输出包）。此时会弹出一个窗口，在显示窗口的底部将显示有这个包的信息和以十六进制形式发送的数据，如图8-29所示。注意，数据由以下4个字节组成：454

```
50  3D  03  54
P  =3T
```

这对应于ASCII字符串P=3T。这是从PC发送给微控制器的实际包。

USBTrace也能详细地显示设备的描述符，如图8-29中屏幕的下方所示。455

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

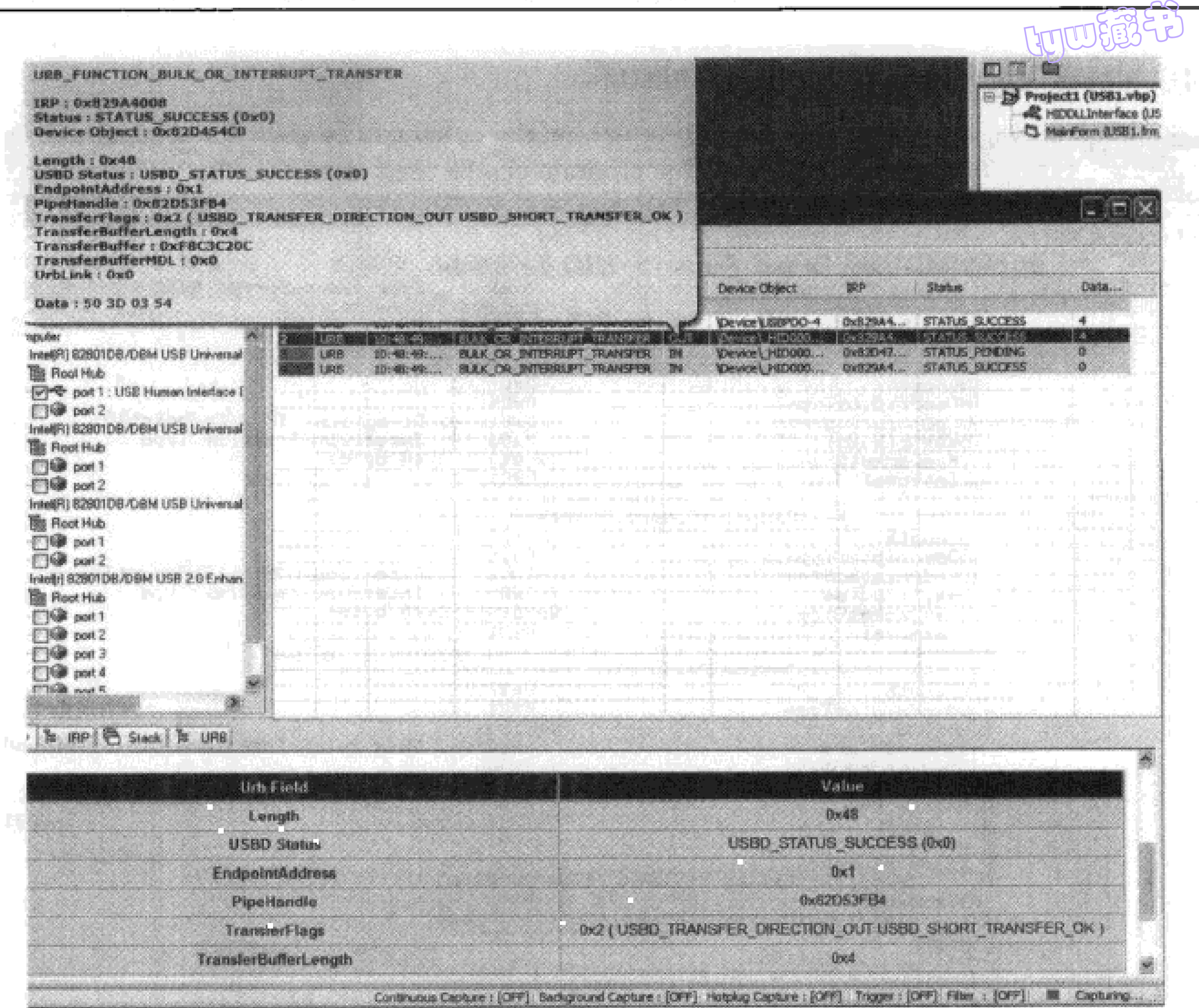


图8-29　显示包的内容

使用mikroC的HID终端

mikroC IDE提供了一个USB终端接口，用来在USB总线上收发数据。该程序可以代替VISUAL BASIC程序，用来检测USB接口。其操作步骤如下。

■ 在mikroC IDE中，选择Tool → HID Terminal。

■ 将微控制器连接到PC的USB端口。

■ 在HID设备里查看产品ID：

- 为了点亮LED灯0、1、4和5，在Communication表单处输入P=3T，然后点击Send按钮，如图8-30所示（注意，数值3的ASCII码值的位模式为0011 0011）
- 此后，微控制器的LED灯0、1、4和5将被点亮

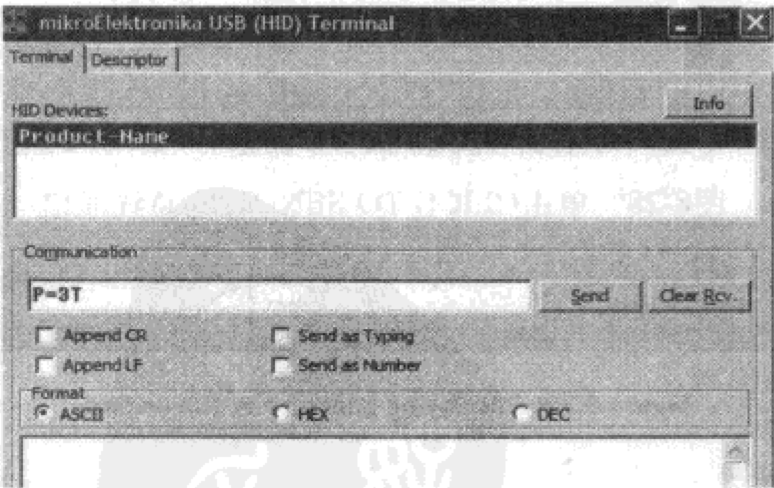


图8-30　使用HID终端发送数据到USB设备

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

项目 8.2 基于 USB 的微控制器的输入输出

456

这个项目与项目8.1很相似，不同之处在于项目8.2是双向通信的，而在项目8.1中要通过在PORTB输出数据来发送到微控制器。另外，PORTB数据是从微控制器接收而来的，并将在PC上显示出来。

PC向微控制器发送两条命令。

- 命令P=*n*T，要求微控制器发送数据字节*n*到PORTB。
- 命令P=??，要求微控制器读取自身的PORTB数据，并以字节形式发送给PC。然后PC在屏幕上显示这些数据。微控制器以熟悉的格式P=*n*T发送数据。

该项目的硬件与上一个项目的硬件是相同的，如图8-11所示，其中微控制器工作在8 MHz晶振下，8个LED灯被连接到PIC18F4550微控制器的PORTB。

本项目中只使用一个窗口界面，如图8-31所示，窗口的上部分与项目8.1中的相同，即向微控制器的PORTB发送数据。窗口中还放置有一个文本框和一个CLICK TO RECIEVE按钮。点击该按钮，PC将发送命令P=??给微控制器。微控制器读取PORTB数据并以格式P=*n*T发送给PC，PC收到数据并将其显示在文本框里。

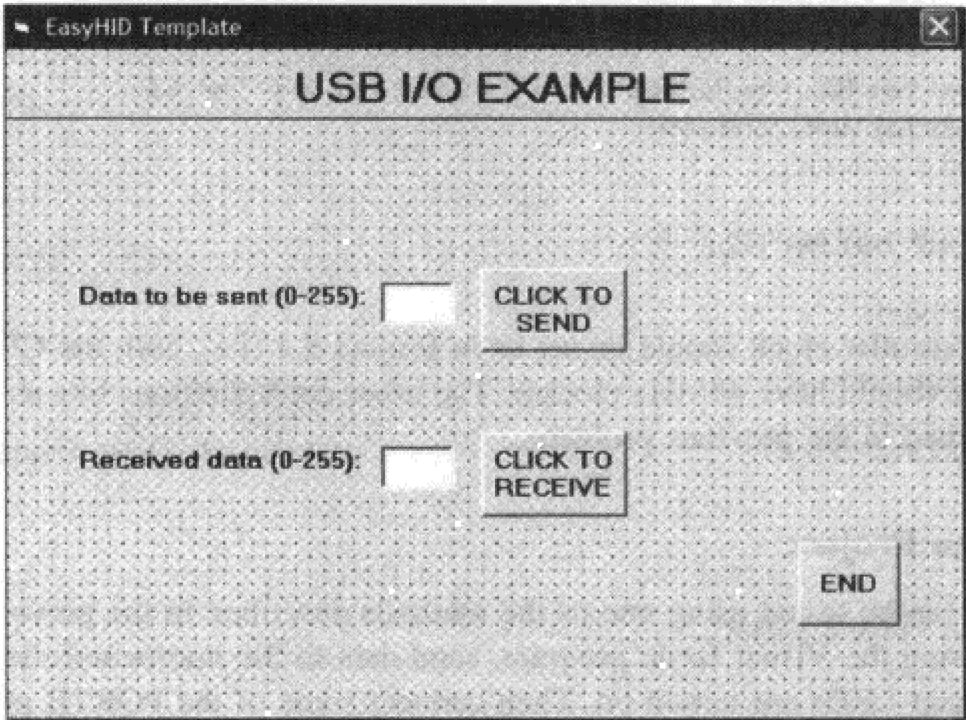


图8-31 项目的VISUAL BASIC窗口

457

图8-32给出了项目的mikroC程序。程序名为USB2.C，与前面项目的程序很相似。但是在这里，当微控制器接收到来自PC的P=??命令时，微控制器读取PORTB数据并使用mikroC函数`Hid_Write`将数据以该格式发送给PC。

```
/***************************************************************************
           USB BASED MICROCONTROLLER INPUT/OUTPUT PORT
           ==========================================

In this project a PIC18F4550 type microcontroller is connected
to a PC through the USB link.

A Visual Basic program runs on the PC where the user enters the
bits to be set or cleared on PORTB of the microcontroller. The
PC sends a command to the microcontroller requesting it to set
or reset the required bits of the microcontroller PORTB. In addition,
```

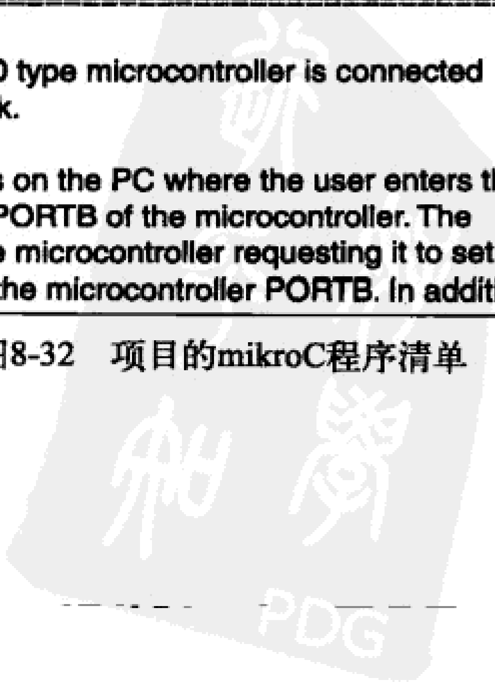


图8-32 项目的mikroC程序清单

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
the PORTB data can be requested from the microcontroller and displayed
on the PC.

The microcontroller is operated from a 8MHz crystal, but the CPU
clock frequency is increased to 48MHz. Also, the USB module operates
with 48MHz.

The commands are:

From PC to microcontroller: P=nT (Send data byte n to PORTB)
                            P=?? (Give me PORTB data)

From microcontroller to PC: P=nT (Here is my PORTB data)

Author:     Dogan Ibrahim
Date:       September 2007
File:       USB2.C
*****************************************************************************/

#include "C:\Program
Files\Mikroelektronika\mikroC\Examples\EasyPic4\extra_examples\HID-
library\USBdsc.c"

unsigned char Read_buffer[64];
unsigned char Write_buffer[64];
unsigned char num,i;
//
// Timer interrupt service routine
//
void interrupt()
{
  HID_InterruptProc();                     // Keep alive
  TMR0L = 100;                             // Reload TMR0L
  INTCON.TMR0IF = 0;                       // Re-enable TMR0 interrupts
}


//
// Start of MAIN program
//
void main()
{

   ADCON1 = 0xFF;                  // Set PORTB to digital I/O
   TRISB = 0;                      // Set PORTB to outputs
   PORTB = 0;                      // PORTB all 0s to start with

//
// Set interrupt registers to power-on defaults
// Disable all interrupts
//
   INTCON=0;
   INTCON2=0xF5;
   INTCON3=0xC0;
   RCON.IPEN=0;
   PIE1=0;
   PIE2=0;
   PIR1=0;
   PIR2=0;
//
```

图8-32 （续）

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
// Configure TIMER 0 for 20ms interrupts. Set prescaler to 256
// and load TMR0L to 156 so that the time interval for timer
// interrupts at 8MHz is 256*156*0.5 = 20ms
//
// The timer is in 8-bit mode by default
//
 T0CON  = 0x47;                    // Prescaler = 256
 TMR0L  = 100;                     // Timer count is 256-156 = 100
 INTCON.TMR0IE = 1;                // Enable T0IE
 T0CON.TMR0ON = 1;                 // Turn Timer 0 ON
 INTCON = 0xE0;                    // Enable interrupts

//
// Enable USB port
//
  Hid_Enable(&Read_buffer, &Write_buffer);
  Delay_ms(1000);
  Delay_ms(1000);
//
// Read from the USB port. Number of bytes read is in num
//

  for(;;)                          // do forever
{
    num=0;
    while(num != 4)
    {num = Hid_Read();
    }
    if(Read_buffer[0] == 'P' && Read_buffer[1] == '=' &&
     Read_buffer[2] == '?' && Read_Buffer[3] == '?')
    {
      TRISB = 0xFF;
      Write_buffer[0] = 'P';   Write_buffer[1] = '=';
      Write_buffer[2] = PORTB; Write_buffer[3] = 'T';
      Hid_Write(&Write_buffer,4);
    }
    else
    {
      if(Read_buffer[0] == 'P' && Read_buffer[1] == '=' &&
       Read_buffer[3] == 'T')
      {
        TRISB = 0;
        PORTB = Read_buffer[2];
      }
    }
 }
  Hid_Disable();

}
```

图8-32 （续）

程序检查接收到的命令格式。对于命令P=??，PORTB将被配置为输入，将PORTB数据读入到Write_buffer[2]，将Write_buffer发送到PC，其中Write_buffer[0]＝P，Write_buffer[1]＝=，Write_buffer[3]＝T。其程序代码如下：

```
if(Read_buffer[0] == 'P' && Read_buffer[1] == '=' &&
Read_buffer[2] == '?' && Read_Buffer[3] == '?')
{
TRISB = 0×FF;
Write_buffer[0] = 'P'; Write_buffer[1] = '='; Write_buffer[2] =
```

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

```
  PORTB; Write_buffer[3] = 'T';
Hid_Write(&Write_buffer,4);
}
```

对于命令P=*n*T，PORTB将被配置为输出，将Read_buffer[2]发送到PORTB，程序如下：

```
if(Read_buffer[0] == 'P' && Read_buffer[1] == '=' &&
Read_buffer[3] == 'T')
{
TRISB = 0;
PORTB = Read_buffer[2];
}
```

微控制器的时钟应按项目8.1进行设置（即CPU和USB模块的时钟频率均为48 MHz）。其他配置也应按前面所描述的进行设置。

测试项目

458～460

项目的测试可以选用前面项目中介绍的方法。如果使用VISUAL BASIC程序，那么发送数据到微控制器，确定LED灯被正确点亮。然后将一些PORTB引脚连接到逻辑0电平，单击CLICK TO RECEIVE按钮。微处理器将读取PORTB数据并发送到PC，PC屏幕上将显示该数据。

也可以使用mikroC IDE 的HID终端来测试项目，其步骤如下。

- 打开HID终端。
- 发送命令（如P=1T）到微控制器，点亮LED灯，确定LED灯被正确点亮（在此例中，LED灯0、4和5应该被点亮，对应数据模式为0011 0001）。
- 将PORTB的位2和3连接到逻辑1电平，将其他的6位接地。
- 发送命令P=??到微控制器。
- PC上将会显示数字12，对应的位模式为0000 1100 。

图8-33给出了该项目的VISUAL BASIC程序清单。这里只给出了主程序，因为库声明与图8-19中的相同。当数据到达USB总线时，程序跳转到子程序OnRead。检测数据的格式是否为P=*n*T，如果格式正确，则将接收到的数据字节显示在文本框里。

```
' vendor and product IDs
Private Const VendorID = 4660
Private Const ProductID = 1

' read and write buffers
Private Const BufferInSize = 8
Private Const BufferOutSize = 8
Dim BufferIn(0 To BufferInSize) As Byte
Dim BufferOut(0 To BufferOutSize) As Byte

Private Sub Command1_Click()
 Form_Unload (0)
End
End Sub

Private Sub Command2_Click()
  BufferOut(0) = 0                    ' first byte is always the report ID
  BufferOut(1) = Asc("P")             ' first data item ("P")
  BufferOut(2) = Asc("=")             ' second data item ("=")
  BufferOut(3) = Val(txtno)           ' third data item (data)
  BufferOut(4) = Asc("T")             ' fourth data item ("T")

  ' write the data (don't forget, pass the whole array)...
```

图8-33　项目的VISUAL BASIC程序清单

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
  hidWriteEx VendorID, ProductID, BufferOut(0)
  lblstatus = "Data sent..."

End Sub
' ***************************************************************************

' Send command P=?? to the microcontroller to request its PORTB data
'****************************************************************************

*
Private Sub Command3_Click()
  BufferOut(0) = 0                    ' first byte is always the report ID
  BufferOut(1) = Asc("P")             ' first data item ("P")
  BufferOut(2) = Asc("=")             ' second data item ("=")
  BufferOut(3) = Asc("?")             ' third data item ("?")
  BufferOut(4) = Asc("?")             ' fourth data item ("?")

  ' write the data (don't forget, pass the whole array)...
  hidWriteEx VendorID, ProductID, BufferOut(0)
  lblstatus = "Data requested..."

End Sub

' ************************************************************************
' when the form loads, connect to the HID controller - pass
' the form window handle so that you can receive notification
' events...
'*************************************************************************
Private Sub Form_Load()
   ' do not remove!
   ConnectToHID (Me.hwnd)
   lblstatus = "Connected to HID..."
End Sub

'*********************************************************************

' disconnect from the HID controller...
'*********************************************************************
Private Sub Form_Unload(Cancel As Integer)
   DisconnectFromHID
End Sub

'*********************************************************************

' a HID device has been plugged in...
'*********************************************************************
Public Sub OnPlugged(ByVal pHandle As Long)
   If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) =
ProductID Then
   lblstatus = "USB Plugged....."
   End If
End Sub

'*********************************************************************

' a HID device has been unplugged...
'*********************************************************************
Public Sub OnUnplugged(ByVal pHandle As Long)
   If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) =
ProductID Then
```

图8-33 （续）

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
    lblstatus = "USB Unplugged...."
    End If
End Sub

'*****************************************************************

' controller changed notification - called
' after ALL HID devices are plugged or unplugged
'*****************************************************************
Public Sub OnChanged()
    Dim DeviceHandle As Long

    ' get the handle of the device we are interested in, then set
    ' its read notify flag to true - this ensures you get a read
    ' notification message when there is some data to read...
    DeviceHandle = hidGetHandle(VendorID, ProductID)
    hidSetReadNotify DeviceHandle, True
End Sub

'*****************************************************************

' on read event...
'*****************************************************************
Public Sub OnRead(ByVal pHandle As Long)
            ' read the data (don't forget, pass the whole array)...
            If hidRead(pHandle, BufferIn(0)) Then
             ' The data is received in the format: P=nT where the first byte
             ' is the report ID. i.e. BufferIn(0)=reportID, BufferIn(0)="P" and so on
             ' Check to make sure that received data is in correct format
              If (BufferIn(1) = Asc("P") And BufferIn(2) = Asc("=") And
                  BufferIn(4) = Asc("T")) Then
               txtreceived = Str$(BufferIn(3))
               lblstatus = "Data received..."
              End If
            End If
        End Sub
```

图8-33 （续）

461～463

在本书附属资源中的文件夹USB2里，附有VISUAL BASIC程序的可安装版可供使用。

项目 8.3 基于 USB 的周围气压 PC 显示

在这个项目中，周围气压传感器被连接到PIC18F4550微控制器，使用USB连接，每隔1秒钟将测量到的气压发送到PC并进行显示。

该项目使用MPX4115A压力传感器。该传感器产生一个模拟电压，与周围的气压成正比。该设备可用的封装有6引脚的和8引脚的。

6引脚封装传感器的引脚配置为：

引脚	描述
1	输出电压
2	接地
3	+5V供电
4～6	未使用

而8引脚封装传感器的引脚配置为：

引脚	描述
1	未使用
2	+5V供电

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

tyw藏书

3 接地
4 输出电压
5 ~ 8 未使用

464

图8-34为两种不同类型引脚配置的传感器的外形图。



图8-34 MPX4115A 压力传感器

该传感器的输出电压由下式决定：

$$V = 5.0 \times (0.009 \times kPa - 0.095) \tag{8.1}$$

或者

$$kPa = \frac{\frac{V}{5.0} + 0.095}{0.009} \tag{8.2}$$

其中，

kPa为大气压（千帕）

V为传感器的输出电压（V）

大气压的测量值通常用毫巴（mb）表示。在海平面，当温度为15℃时，大气压为1 013.3 mb。在式（8.2）中，气压单位是kPa。将kPa转化为mb，必须将式（8.2）乘以10：

$$mb = 10 \times \frac{\frac{V}{5.0} + 0.095}{0.009} \tag{8.3}$$

或者

$$mb = \frac{2.0V + 0.95}{0.009} \tag{8.4}$$

465

图8-35为MPX4115A传感器输出电压随大气压变化的曲线图。本项目重点关注在800 mb ~ 1 100 mb范围内的气压。

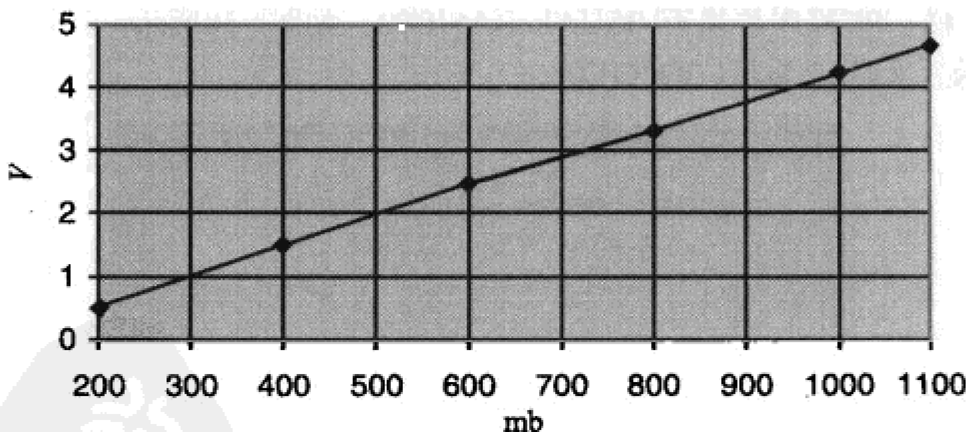


图8-35 传感器输出电压随气压变化的曲线

以mb作单位计算气压的步骤如下：

- 使用微控制器的一个A/D通道，读取气压传感器的输出电压值；
- 使用式（8.4），将电压转化为以mb为单位的气压值。

466

该项目的方框图如图8-36所示。

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

lyw藏书

图8-37所示为该项目的电路原理图。其中传感器的输出端被连接到微控制器的模拟输入端AN0。和项目8.2一样，USB连接器被连接到端口引脚RC4和RC5，微控制器运行在8 MHz晶振下。

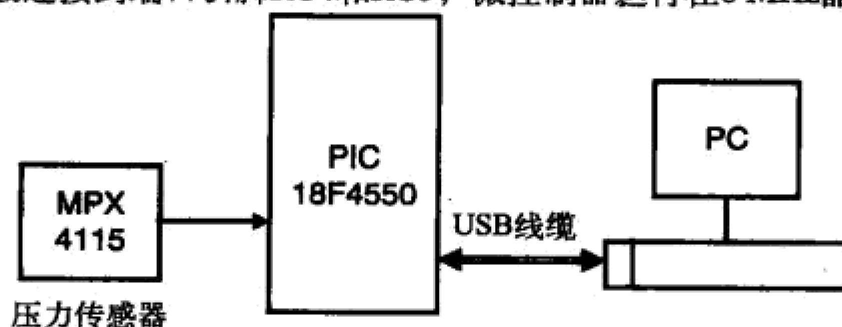


图8-36　项目的方框图

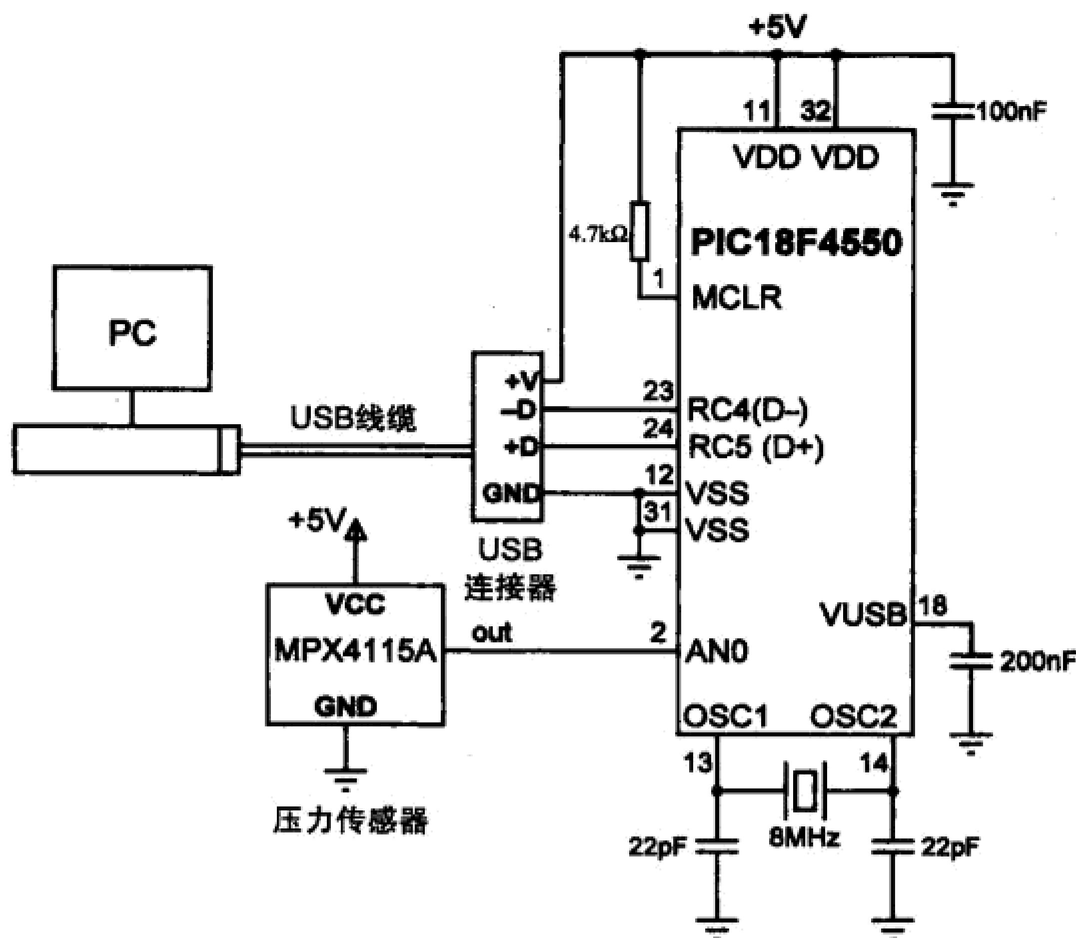


图8-37　项目的电路原理图

和前面的项目一样，PC程序是基于VISUAL BASIC的。如图8-38所示，这里只需要使用一个控制窗口，用来每隔1秒显示一次以mb为单位的气压值。

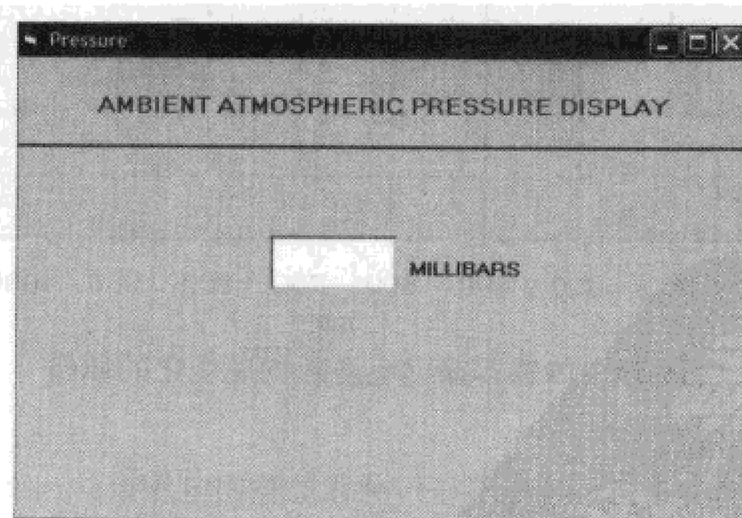


图8-38　显示气压值的VISUAL BASIC窗口

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

图8-39给出了该项目的微控制器程序（名为PRESSURE.C）清单。在主程序的开始处，通过清零ADCON1，将端口PORTA引脚设置为输入，从而将这些端口引脚定义为模拟输入端。然后，将中断寄存器设置为上电复位值。将定时器中断TMR0设置为每3.3 ms产生一次中断以保持USB总线处于激活状态。接下来，使能微控制器的USB端口，并初始化ADCON2，将A/D时钟频率设置为$F_{osc/64}$。 467

```
/******************************************************************************
        USB BASED ATMOSPHERIC PRESSURE DISPLAY ON PC
        ==============================================

In this project a PIC18F4550 type microcontroller is connected
to a PC through the USB link.

In addition, a MPX4115A type pressure sensor IC is connected to analog port AN0 of
the microcontroller. The microcontroller reads the atmospheric presure and sends it to
the PC every second. The PC displays the pressure on the screen.

A Visual Basic program runs on the PC which reads the pressure from the USB port
and then displays it on a form.

The microcontroller is operated from a 8MHz crystal, but the CPU clock frequency is
increased to 48MHz. Also, the USB module operates with 48MHz.

The pressure is sent to the PC in millibars as a 4 digit integer
number.

Author:     Dogan Ibrahim
Date:       September 2007
File:       PRESSURE.C
******************************************************************************/

#include "C:\Program
Files\Mikroelektronika\mikroC\Examples\EasyPic4\extra_examples\HID-
library\USBdsc.c"

unsigned char num,i,j;
unsigned long Vin, Pint;
unsigned char op[12], Pressure[4], Read_buffer[4];
float mV,V,Pmb;

//
// Timer interrupt service routine
//
void interrupt()
{
 HID_InterruptProc();                       // Keep alive
 TMR0L = 100;                               // Reload TMR0L
 INTCON.TMR0IF = 0;                         // Re-enable TMR0 interrupts
}


//
// Start of MAIN program
//
void main()
{
     ADCON1 = 0;                            // Set inputs as analog, Ref=+5V
     TRISA = 0xFF;                          // Set PORT A as inputs
 //
```

图8-39　项目的微控制器程序

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源


```
// Set interrupt registers to power-on defaults
// Disable all interrupts
//
    INTCON=0;
    INTCON2=0xF5;
    INTCON3=0xC0;
    RCON.IPEN=0;
    PIE1=0;
    PIE2=0;
    PIR1=0;
    PIR2=0;
//
// Configure TIMER 0 for 3.3ms interrupts. Set prescaler to 256
// and load TMR0L to 156 so that the time interval for timer
// interrupts at 48MHz is 256*156*0.083 = 3.3ms
//
// The timer is in 8-bit mode by default
//
  T0CON  = 0x47;                               // Prescaler = 256
  TMR0L  = 100;                                // Timer count is 256-156 = 100
  INTCON.TMR0IE = 1;                           // Enable T0IE
  T0CON.TMR0ON = 1;                            // Turn Timer 0 ON
  INTCON = 0xE0;                               // Enable interrupts


//
// Enable USB port
//
  Hid_Enable(&Read_buffer, &Pressure);
  Delay_ms(1000);
  Delay_ms(1000);

//
// Configure A/D converter. AN0 is used in this project
//
    ADCON2 = 0xA6;                             // A/D clock = Fosc/64, 8TAD
//
// Endless loop. Read pressure from the A/D converter,
// convert into millibars and send to the PC over the
// USB port every second
//
  for(;;)                                      // do forever
{
      Vin = Adc_Read(0);                       // Read from channel 0 (AN0)
      mV = (Vin * 5000.0) / 1024.0;            // In mv=Vin x 5000/1024
      V = mV /1000.0;                          // Pressure in Volts
      Pmb = (2.0*V + 0.95) / 0.009;            // Pressure in mb
      Pint = (int)Pmb;                         // As an integer number
      LongToStr(Pint,op);                      // Convert to string in "op"
//
// Remove leading blanks
//
      for(j=0; j<4; j++)Pressure[j]=' ';

      j=0;
      for(i=0;i<=11;i++)
      {
        if(op[i] != ' ')                       // If a blank
        {
```

图8-39 （续）

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
            Pressure[j]=op[i];
            j++;
          }
        }
//
// Send pressure (in array Pressure) to the PC
//
     Hid_Write(&Pressure,4);                    // Send to USB as 4 characters
     Delay_ms(1000);                            // Wait 1 second
 }
   Hid_Disable();

}
```

图8-39 （续）

使用for语句构造一个无限循环。在这个循环内，将气压传感器的数据读入到变量Vin中，然后将其转化为物理电压（单位是mV）并存储在变量mV中。然后使用式（8.4）计算出大气压，并存储在长整型变量Pint中。使用mikroC函数LongToStr，将这些整数转化为字符串存储在数组op里。在这个数组中，清除开头的空格，将得到的结果存储在字符数组Pressure中。然后调用mikroC的USB函数Hid_Write将气压以4字符数据的格式发送到USB总线。程序在等待1秒后，不断重复上述过程。

使用一个8 MHz的晶振来为微控制器提供时钟脉冲。微控制器CPU和USB模块运行在48 MHz时钟率下，时钟和配置寄存器的设置和本章中其他项目是一样的。

468
~
470

基于VISUAL BASIC的程序被命名为PRESSURE。子程序OnRead用来接收到达PC的USB端口的数据，然后将其显示在屏幕窗口中。程序并不向USB总线发送任何数据。图8-40给出了程序清单（不包括全局变量声明）。

```
' vendor and product IDs
Private Const VendorID = 4660
Private Const ProductID = 1

' read and write buffers
Private Const BufferInSize = 8
Private Const BufferOutSize = 8
Dim BufferIn(0 To BufferInSize) As Byte
Dim BufferOut(0 To BufferOutSize) As Byte

Private Sub Command1_Click()
 Form_Unload (0)
End
End Sub

' ***********************************************************************

' when the form loads, connect to the HID controller - pass
' the form window handle so that you can receive notification
' events...
'************************************************************************
Private Sub Form_Load()
  ' do not remove!
  ConnectToHID (Me.hwnd)
  lblstatus = "Connected to HID..."
End Sub

'************************************************************************

' disconnect from the HID controller...
```

图8-40 项目的VISUAL BASIC程序

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
'******************************************************************
Private Sub Form_Unload(Cancel As Integer)
  DisconnectFromHID
End Sub

'******************************************************************

' a HID device has been plugged in...
'******************************************************************
Public Sub OnPlugged(ByVal pHandle As Long)
  If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) =
ProductID Then
  lblstatus = "USB Plugged....."
  End If
End Sub
******************************************************************
' a HID device has been unplugged...
'******************************************************************
Public Sub OnUnplugged(ByVal pHandle As Long)
  If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) =
ProductID Then
  lblstatus = "USB Unplugged...."
  End If
End Sub

'******************************************************************

' controller changed notification - called
' after ALL HID devices are plugged or unplugged
'******************************************************************
Public Sub OnChanged()
  Dim DeviceHandle As Long

  ' get the handle of the device we are interested in, then set
  ' its read notify flag to true - this ensures you get a read
  ' notification message when there is some data to read...
  DeviceHandle = hidGetHandle(VendorID, ProductID)
  hidSetReadNotify DeviceHandle, True
End Sub

'******************************************************************

' on read event...
'******************************************************************
Public Sub OnRead(ByVal pHandle As Long)
  Dim pressure As String

  If hidRead(pHandle, BufferIn(0)) Then
    ' The first byte is the report ID. i.e. BufferIn(0)=reportID
    pressure = Chr(BufferIn(1)) & Chr(BufferIn(2)) & Chr(BufferIn(3)) &
Chr(BufferIn(4))
    txtno = pressure
  End If
End Sub
```

图8-40　(续)

471～473

图8-41所示为VISUAL BASIC程序的常见输出窗口，用来显示气压值。

在本书附属资源中的文件夹PRESSURE里，附有VISUAL BASIC程序的可安装版供使用。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

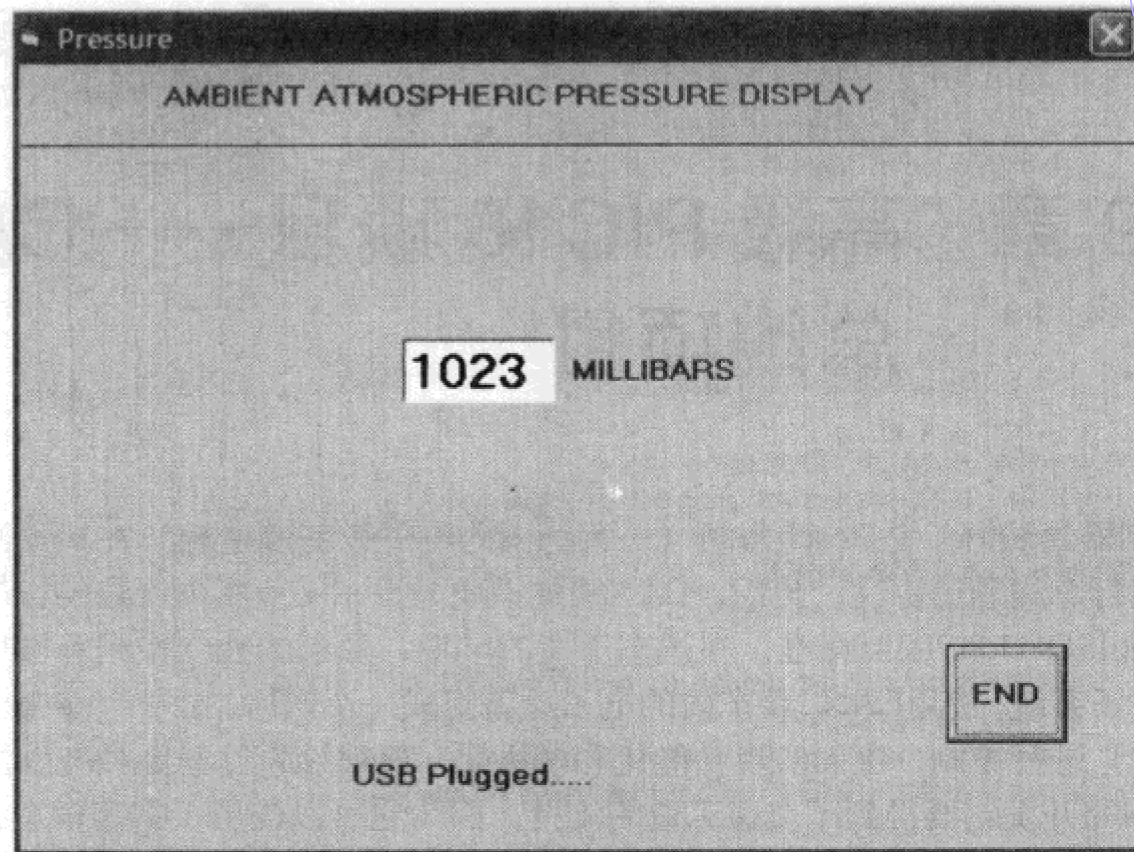


图8-41 VISUAL BASIC程序的常见输出窗口

474

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

第 9 章　高级 PIC18 项目——CAN 总线项目

CAN（控制器局域网）是Bosch公司（一家德国的电子设备制造商）在20世纪80年代初开发出来的一个串行总线通信协议。此后，CAN被建设成为汽车工业车内网络的标准协议，进而成为ISO-11898和ISO-11519国际标准。在汽车工业的早期，局部的独立控制器被用来管理各种执行器和机械电子系统。使用CAN，将车内的电子设备组成一个网络，由一个中央节点（即ECU，发动机控制单元）进行控制，进而实现功能化和模块化，使得诊断过程更加高效。

早期，CAN的开发主要得到了交通工业的支持，因为它可以应用于乘客轿车、轮船、卡车以及其他类型的交通工具。今天，CAN协议被应用于许多要求网络化嵌入式控制的其他领域，包括工业自动化、医疗应用、楼宇自动化、纺织机械和生产机械。在实时应用中的传感器、执行器、控制器以及其他节点之间，CAN提供了一种高效的通信协议，并以简单性、可靠性、高性能而著称。

CAN协议是基于总线拓扑结构的，通过CAN总线的通信只需要两根线。这种总线具有一个多主机的结构，总线上的每个设备能发送或接收数据。在任何时刻，只有一个设备能发送数据，而其他的设备都只能侦听。如果两个或两个以上设备试图同时发送数据，那么拥有最高优先级的设备将被允许发送它的数据，而其他的设备则返回到接收模式。

如图9-1所示，在一个典型汽车应用中，通常有一条以上的CAN总线，它们的工作速度是不

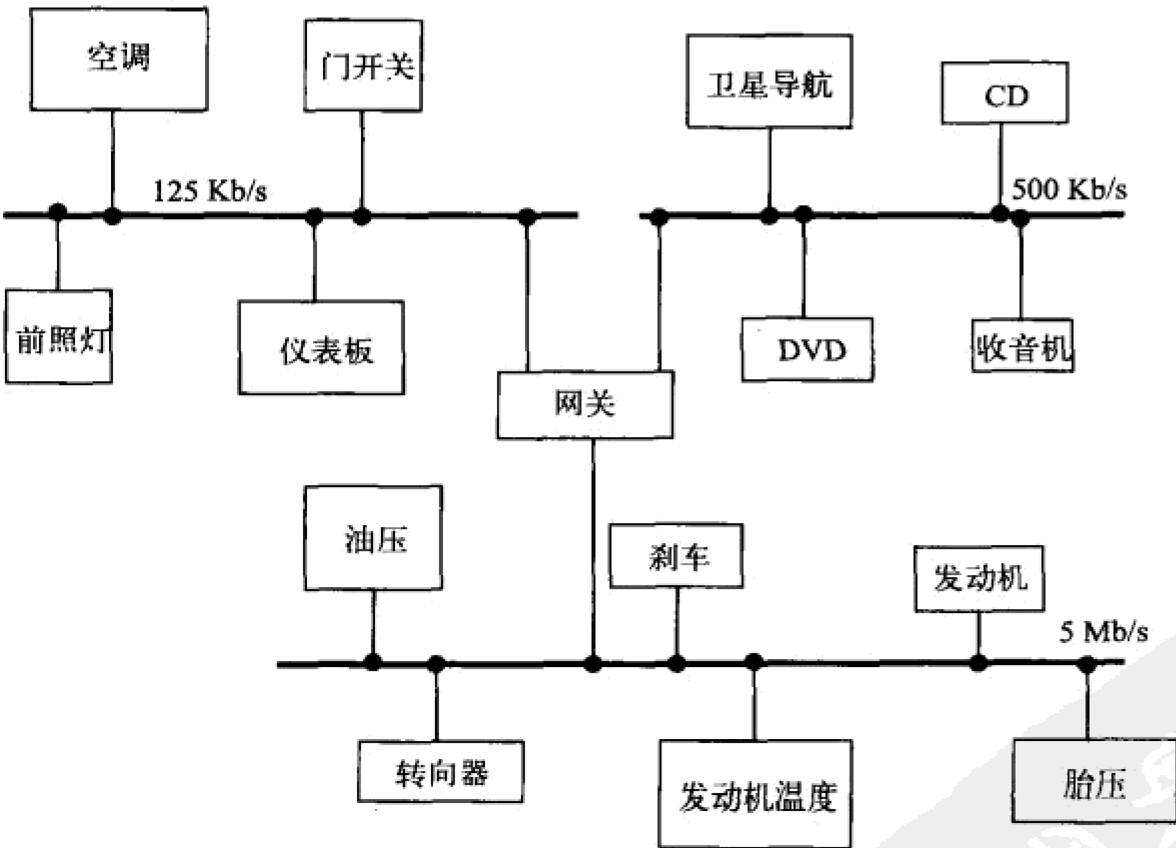


图9-1　汽车中常见的CAN总线应用

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

同的。对于速度较慢的设备，如门控制、气候控制和驾驶信息模块，可以使用一条慢速总线进行连接。对于要求快速响应的设备，如ABS防抱死制动系统、发送控制模块和电子节气门模块，可以使用一条速度较快的CAN总线进行连接。

CAN在汽车工业的应用引发了CAN控制器的大量生产。目前，CAN模块每年的销售量估计为4亿个，许多微控制器（包括PIC微控制器）都集成有CAN控制器，而且售价较低。

图9-2显示了带有3个节点的CAN总线。CAN协议是基于CSMA/CD+AMP（具有报文优先级裁决的载波监听多路访问/冲突检测）协议的，该协议与Ethernet局域网中使用的协议有些相似。 476 当Ethernet网检测到冲突时，发送节点只是停止发送，在等待一段随机的时间后，再次发送；而CAN协议使用仲裁原则来解决冲突问题，只赋予最高优先级的节点发送数据。

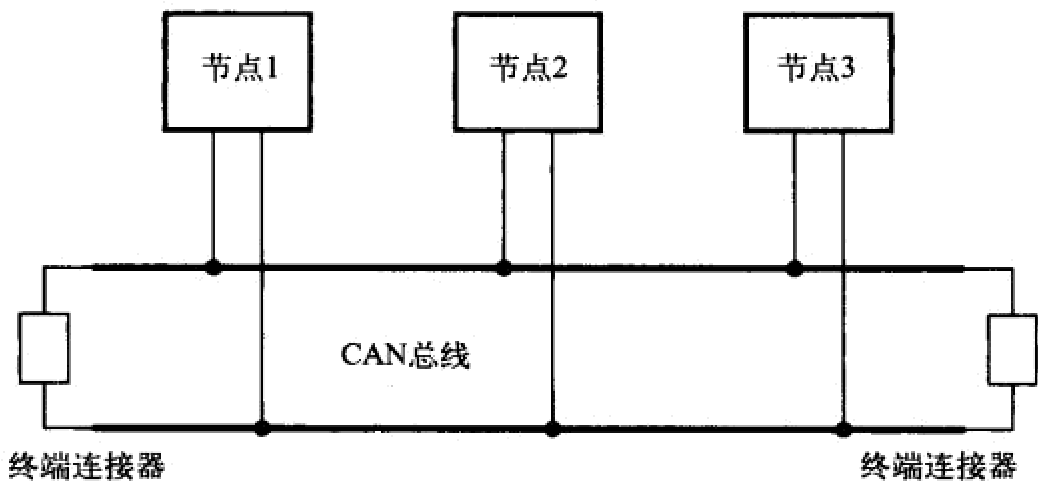


图9-2　CAN总线例子

基本上说，有两种类型的CAN协议：2.0A和2.0B。CAN 2.0A是早期的标准，使用11位的识别码，而CAN 2.0B是新的扩展标准，使用29位的识别码。2.0B控制器能完全向后兼容2.0A控制器，能以任一格式进行接收和发送数据。

2.0A控制器有两种类型。第一种类型的2.0A控制器只能发送和接收2.0A报文，当接收到2.0B的报文时将标志为错误。第二种类型的2.0A微控制器（又被称为2.0B passive）可以发送和接受2.0A报文，但也会应答接收2.0B报文，然后忽略它们。

CAN总线的部分特点如下。

- CAN总线是多主机的。当总线空闲时，任何连接到总线的设备都能开始发送信息。
- CAN总线协议是灵活的。连接到总线的设备是没有地址的，这意味着报文不是基于地址从一个节点发送到另一个节点。取而代之的是，系统内的所有节点都接收总线上传送的每一个报文，由每一个节点自己决定是保存还是放弃接收到的报文。一个单独的报文是 477 发给特定的节点还是许多的节点，这由系统的设计决定。没有地址的另一个优点是，当一个设备挂接到总线或者从总线上移除时，不需要对配置数据作任何的改变（即总线是“热拔插的”）。
- CAN总线提供了远程发送请求（RTR），这意味着总线上的一个节点能向其他节点请求报文。因此，不是等待节点连续地发送报文，而是向节点发送一条报文请求。例如，在汽车内，发动机温度是一个重要的参数，可以将系统设计成通过总线周期性发送温度值。然而，一个更好的解决方案是当需要时才请求温度数据，因为它在维持网络完整性的前提下将总线流量降低到最小。
- CAN总线的通信速度不是固定的。可以为连接到总线上的设备，设置任意的通信速度。
- 所有连接到总线的设备都能检测到错误。设备检测到错误后立即通知其他的设备。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

■ 多个设备可以同时连接到总线，而且连接到总线的设备数量是没有逻辑限制的。但实际上，连接到总线的设备数量是受总线的延迟时间和电气负载限制的。

CAN总线上的数据是差分形式的，并且分为两个状态：显性和隐性。图9-3显示了总线上的电压状态。总线将逻辑位0定义为显性位，将逻辑位1定义为隐性位。当总线上有仲裁协议时，478 显性位状态总能胜过隐性位状态。在隐性状态下，差分电压CANH和CANL都低于最小的阈值(即低于0.5 V的接收器输入和低于1.5 V的发送器输出)。在显性状态下，差分电压CANH和CANL都大于最小的阈值。

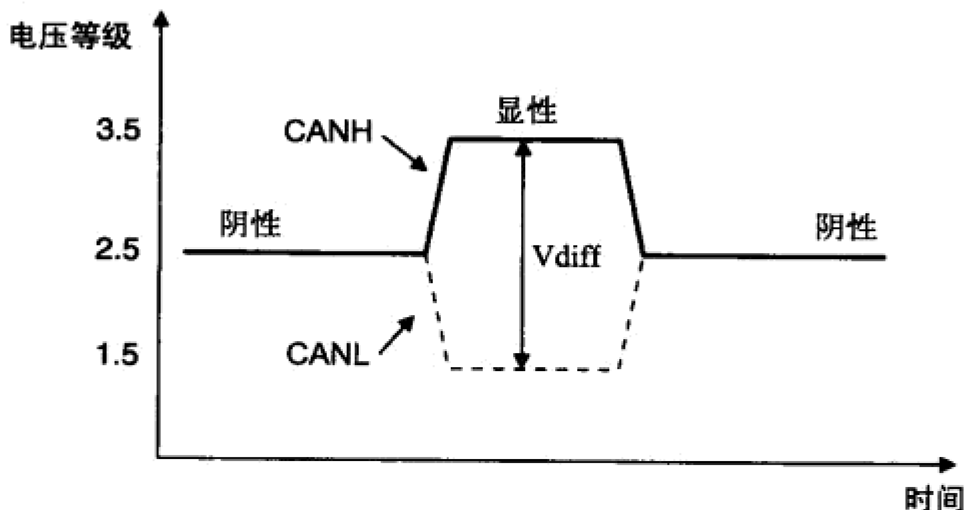


图9-3　CAN逻辑状态

ISO-11898 CAN总线规定了连接在总线上的设备必须能以1 Mb/s的速度驱动40 m长的线缆。通常，通过降低总线速度，可以驱动更长的总线。图9-4显示了总线长度关于通信速度的变化曲线。例如，使用一条1 000 m长的总线，可以得到的最大速度为40 Kb/s。

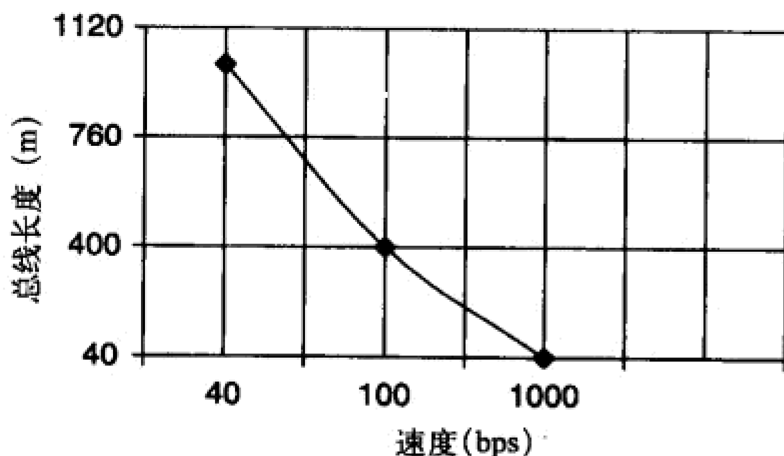


图9-4　总线速度和总线长度

对CAN总线进行终止，可将总线上的信号反射降到最小程度。ISO-11898要求总线有120 Ω的特征阻抗。可以使用以下的方法终止总线：

■ 标准终止；
■ 分裂终止；
■ 加偏压的分离终止。

在标准终止中，最普通的终止方法是在在总线的每一个末端使用一个120 Ω的电阻，如图9-5a所示。在分裂终止中，将总线末端分裂，并使用单个的60 Ω电阻，如图9-5b所示。分裂终止考虑了降低信号发射，该方法正变得日渐流行。479 加偏压的分裂终止有些类似于分裂终止，只不过它需要在总线末端使用一个电压分压器和一个电容器。这种方法提高了总线的EMC性能，如图9-5c所示。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

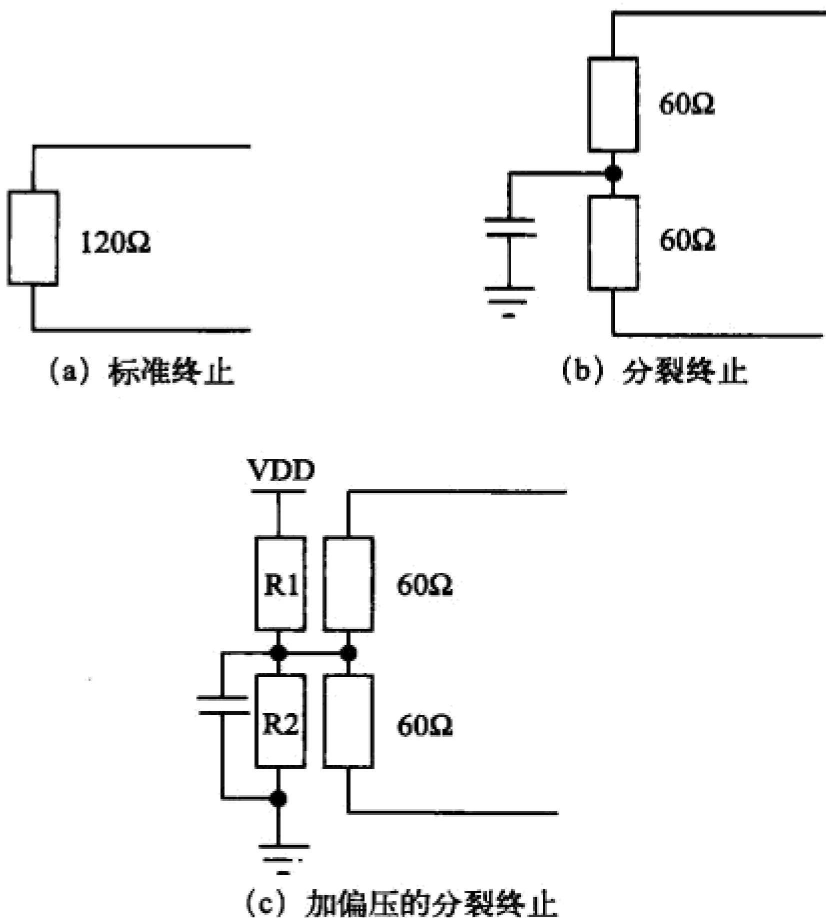


图9-5 总线终止方法

许多的网络协议都使用7层开放系统互连（OSI）模型来描述。CAN协议包括OSI参考模型的数据链路层、物理层，如图9-6所示。数据链路层（DLL）由逻辑链接控制（LLC）和媒体访问控制（MAC）组成。LLC管理过载通知、接收滤波和恢复管理。MAC管理数据封装、帧编码、错误检测和数据的序列化/反序列化。物理层包括物理信令层（PSL），物理介质关联（PMA）和介质相关接口（MDI）。PSL管理位的编码/解码和位时序。PMA管理驱动器/接收器的特征，而MDI是指连接和线缆。 480

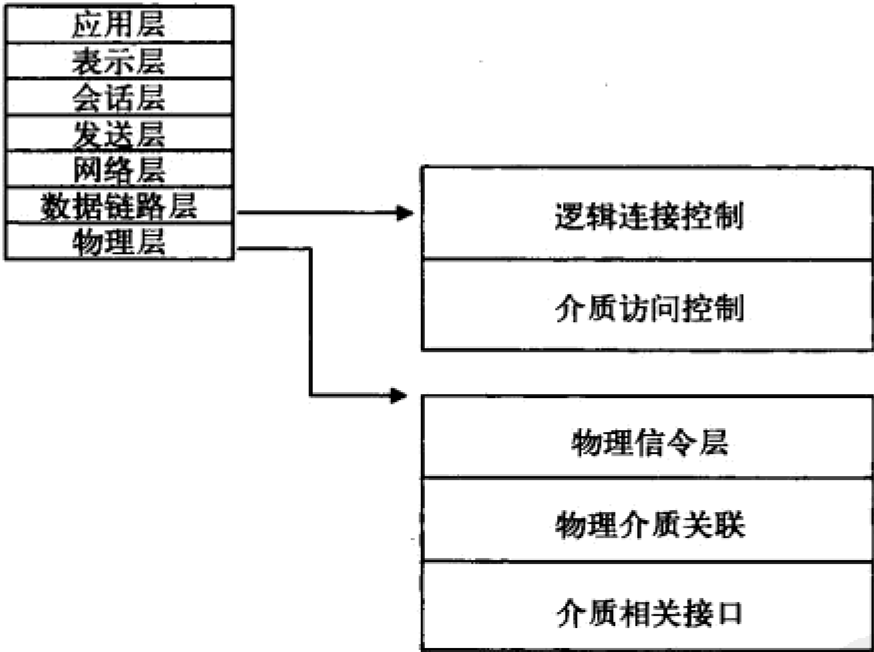


图9-6 CAN和OSI模型

在CAN协议中，基本上有4种类型的报文帧：数据帧、遥控帧、错误帧和过载帧。数据帧和遥控帧需要用户进行设置。另外两种类型的报文帧则由CAN硬件进行设置。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

9.1 数据帧

数据帧有两种格式：标准格式（有11位ID）和扩展格式（有29位ID）。发送设备使用数据帧来将数据发送到接收设备，数据帧是用户处理的最重要的帧。图9-7显示了数据帧的结构。一个标准的数据帧从帧起始位起，紧接着是11位的识别码和远程发送请求（RTR）位。识别码和RTR形成了12位的仲裁字段。控制字段是6位宽的，用来说明数据字段中的字节数量。数据字段可以是0 B～8 B。在数据字段之后是CRC字段，用来检查接收到的位序列是否被破坏。ACK字段是2位宽的，被发送器用来从接收器接收有效帧的应答报文。报文末端是一个7位的帧结束字段（EOF）。在扩展数据帧中，仲裁字段是32位宽的（29位的识别码+1位的IDE（用来将报文定义为扩展数据帧）+1位没有被使用的SSR+1位的RTR），如图9-8所示。

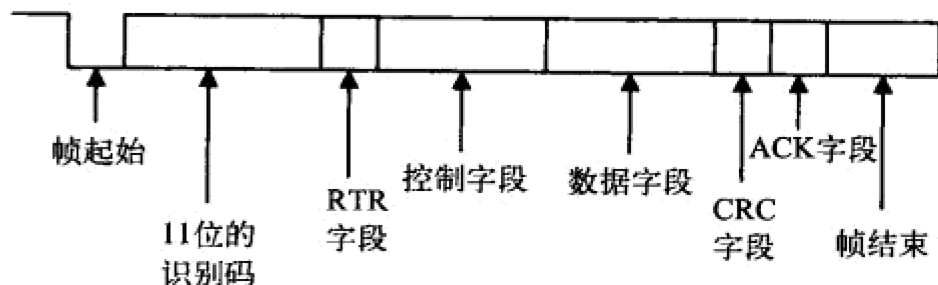


图9-7 标准的数据帧

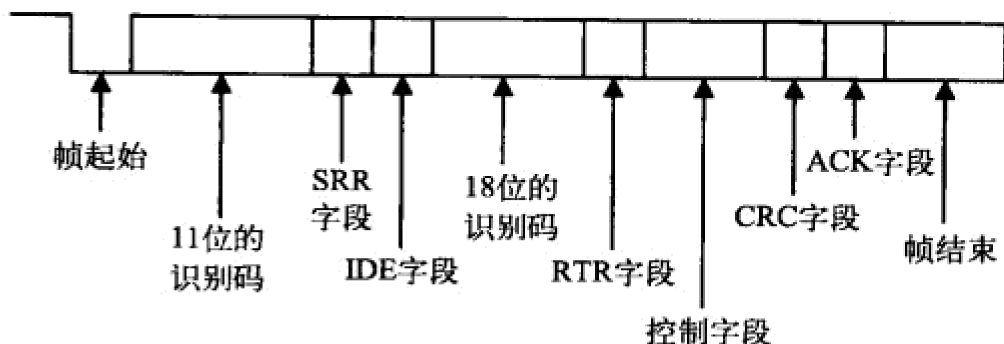


图9-8 扩展数据帧

数据帧由以下的字段组成。

9.1.1 帧起始（SOF）

帧起始字段用来表示一个数据帧的开始，在标准格式和扩展格式中都有帧起始。

9.1.2 仲裁字段

仲裁用来解决当几个设备同时在总线上发送报文时产生的总线冲突。仲裁字段用来指明帧的优先级，它在标准格式和扩展格式下是不同的。在标准格式中有11位，可以设置多达2 032个ID。扩展格式的ID由11个基本ID外加18个扩展ID构成。最多可设置$2\,032\times2^{18}$个离散ID。

在仲裁阶段，每一个发送设备在总线上发送它的识别码，并比较其优先级。如果级别相同，那么设备继续发送。如果设备在总线上检测到一个显性级别，并且正在尝试发送隐性级别，那么该设备会放弃发送而变为一个接收设备。在仲裁总线上只有一个发送器后，该发送器将继续发送它的控制字段、数据字段和其他字段。

图9-9举例说明了仲裁的过程。该例由带有识别码的3个节点构成。

节点1: 111000110011 节点2: 11100111111 节点3: 11100110001

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

假设隐性级别对应于1，而显性级别对应于0，仲裁过程如下执行。

- 所有节点同时开始发送，首先发送SOF位。
- 然后它们发送各自的识别码。节点2的第8位是隐性状态，而节点1和节点3的相应位是显性状态。因而，节点2停止发送，并返回到接收模式。接收期间使用灰色字段表示。
- 节点1的第10位是隐性状态，而节点3的相同位是显性状态。因而，节点1停止发送，并返回到接收模式。
- 现在，总线上只剩下节点3，它可以自由地发送控制字段和数据字段。

注意，总线上的设备是没有地址的。取而代之的是，所有设备在总线上获取所有的数据，每一个节点必须过滤掉它不需要的报文。

483

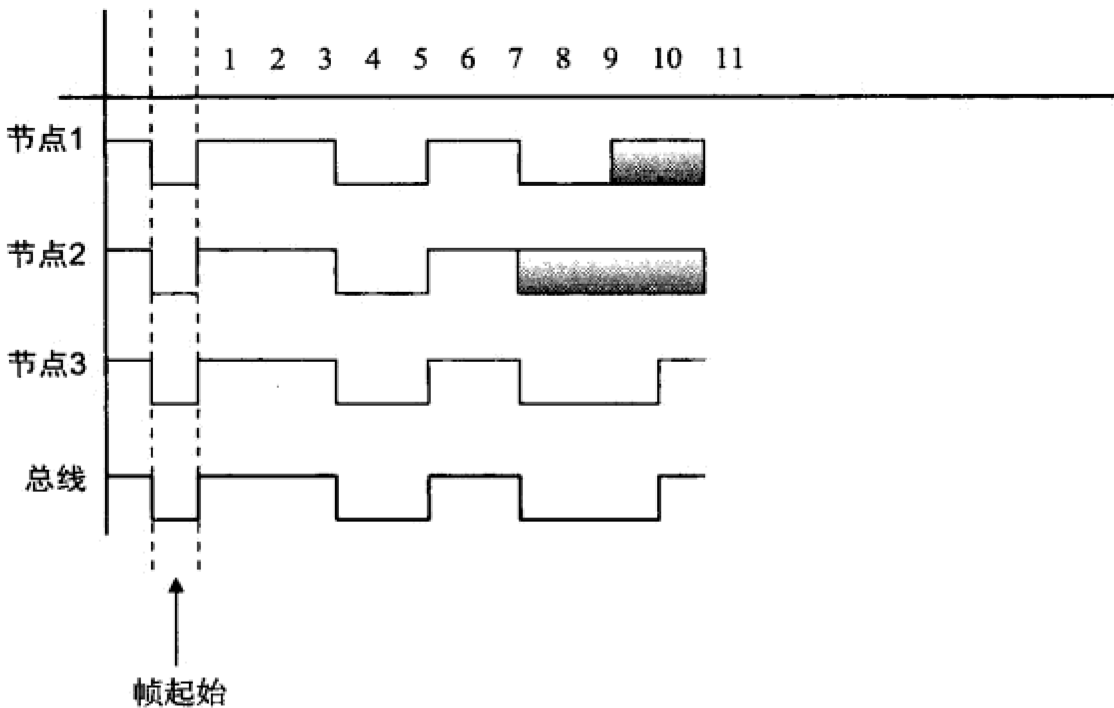


图9-9 CAN总线仲裁实例

9.1.3 控制字段

控制字段是6位宽的，由2个保留位和4个数据长度代码（DLC）位构成，用来说明报文中要发送的数据字节数量。该字段的编码，如表9-1所示，使用6位最多可以编码8个发送字节。

表9-1 编码控制字段

数据字节的编号	DLC3	DLC2	DLC1	DLC0
0	显性级别	显性级别	显性级别	显性级别
1	显性级别	显性级别	显性级别	隐性级别
2	显性级别	显性级别	隐性级别	显性级别
3	显性级别	显性级别	隐性级别	隐性级别
4	显性级别	隐性级别	显性级别	显性级别
5	显性级别	隐性级别	显性级别	隐性级别
6	显性级别	隐性级别	隐性级别	显性级别
7	显性级别	隐性级别	隐性级别	隐性级别
8	隐性级别	显性级别或隐性级别	显性级别或隐性级别	显性级别或隐性级别

9.1.4 数据字段

数据字段承载着报文的实际内容。数据长度从0 B～8 B不等。MSB首先发送。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

9.1.5 CRC字段

CRC字段由一个15位的CRC序列和一个1位的CRC界定符构成，用来检查帧的发送错误。CRC计算包括帧起始、仲裁字段、控制字段和数据字段。比较计算出来的CRC序列和接收到的CRC序列，如果两者不匹配，则认为发生错误。

484

9.1.6 ACK字段

ACK字段用来表明帧被正常的接收。该字段包括2位，一位用作ACK信号，另一个用作ACK界定符。

9.2 遥控帧

接收单元使用遥控帧来从发送单元请求报文发送。遥控帧包括6个字段（如图9-10所示）：帧起始、仲裁字段、控制字段、CRC字段、ACK字段和帧结束。除了没有数据字段以外，遥控帧与数据帧相同。

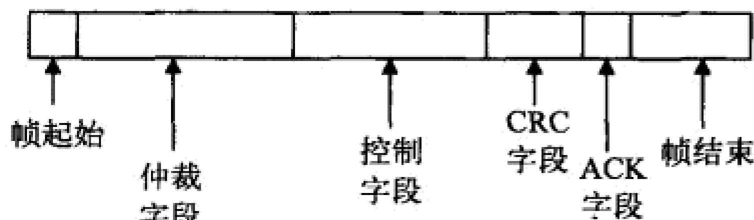


图9-10 遥控帧

9.3 错误帧

错误帧由CAN硬件产生和发送，用来表明在发送时错误发生的时间。错误帧包括一个错误标志和一个错误界定符。错误标志有两种类型：主动错误（包括6个显性位）和被动错误（包括6个隐性位）。错误界定符包括8个隐性位。

9.4 过载帧

接收单元使用过载帧来表明它还没有准备好接收帧。这个帧包括一个过载标志和一个过载界定符。过载标志包括6个显性位，它和错误帧的主动错误标志有着相同的结构。过载界定符包括8个隐性位，它和错误帧的错误界定符有着相同的结构。

485

9.5 位填充

CAN总线利用一种叫作位填充的技术来周期性地同步发送和接收操作，以防止接收节点之间的时序错误。在同样的电平持续5个位之后，则向序列添加一个位的反型数据。在传送数据帧或遥控帧的过程中，如果在从帧起始到CRC序列的任何位置发生同样的电平持续5个位的情况，那么在下一个位（即第6位）插入一个位的反型数据。在接收一个数据帧或遥控帧的过程中，如果在从帧起始到CRC序列的任意位置出现同样的电平持续5个位的情况，那么下一个位（即第6

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

位）将会从接收帧中删除。如果删除的第6位与第5位是相同的电平，则说明检测到一个错误（填充错误）。

9.6 错误类型

CAN总线能识别5种类型的错误：

- 位错误
- CRC错误
- 格式错误
- ACK错误
- 填充错误

当输出电平与总线上的数据电平不匹配时，则会检测到位错误。发送和接收单元都能检测到位错误。CRC错误只会被接收单元检测到。如果从接收到的报文计算出的CRC和接收到的CRC不匹配，则会检测到CRC错误。当一个不合法的格式被检测到时，发送或接收单元则会检测到格式错误。如果ACK字段被发现是隐性的，那么ACK错误只会被发送单元检测到。在填充地段内，当检测到连续6个位具有相同电平时，则表明检测到填充错误。发送和接收单元都可以检测到这种错误。

9.7 标称位时序

CAN总线标称位速率被定义为在非同步的情况下每秒钟发送的位数。标称位速率的倒数就是标称位时间。总线上的所有设备都必须使用相同的位速率，尽管每一个设备自己有着不同的时钟频率。一条报文位包括4个非重叠的时间段：

486

- 同步段（Sync_Seg）
- 传播时间段（Prop_Seg）
- 相位缓冲段1（Phase_Seg1）
- 相位缓冲段2（Phase_Seg2）

Sync_Seg段用来同步总线上的不同节点，电平边沿会出现在该段中。Prop_Seg段用来补偿网络内的物理延迟时间。Phase_Seg1和Phase_Seg2段用来补偿边沿相位误差。这些段可通过同步来加以延长或缩短。采样点是指实际电平的位值所在的点，通常出现在Phase_Seg1末端。将CAN控制器配置成采样3次，使用一个强函数来确定实际的位值。

每段又由可称为时间量子（time quantum）或T_Q的单元组成。通过调整一条报文位包含的T_Q数量和每一字段包含的T_Q数量，可以来设置期望的位时序。T_Q是一个固定单位，由振荡周期得到，每一段的时间量子从1到8个不等。各种时间段的长度分别为：

- Sync_Seg是1个时间量子长度；
- Prop_Seg是可编程的，有1～8个时间量子长度；
- Phase_Seg1是可编程的，有1～8个时间量子长度；
- Phase_Seg2是可编程的，有2～8个时间量子长度。

通过设置位时序，可以对采样点进行设置，因而总线上的多个单元能在同样时序下采样报文。

标称位时序是可编程的，长度为从最小的8个时间量子到最大的25个时间量子不等。通过定

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

义，最小的标称位时间是1 μs，相应地，最大标称位速率为1 Mb/s。标称位时间（T_{BIT}）的计算公式为：

487

$$T_{BIT}=T_Q \times (Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2) \tag{9.1}$$

并且，标称位速率（*NMR*）为：

$$NBR=1/T_{BIT} \tag{9.2}$$

时间量子可从振荡周期和可编程的波特率预分频器（其值为1～64之间的整数）推导而来。时间量子可以表示为：

$$T_Q=2 \times (BRP+1) / F_{OSC} \tag{9.3}$$

其中，T_Q的度量单位是μs，F_{OSC}的度量单位是MHz，BRP是波特率预分频器（0～63）。式（9.2）又可以写成

$$T_Q=2 \times (BRP+1) \times T_{OSC} \tag{9.4}$$

其中，T_{OSC}的度量单位是μs。

以下是一个计算标称位速率的例子。

例9.1 假设时钟频率为20 MHz，波特率预分频器值为1，标称位时间$T_{BIT}=8 \times T_Q$，试确定标称位速率。

解 根据式（9.3），

$$T_Q=2 \times (1+1)/20=0.2\ \mu s$$

且

$$T_{BIT}=8 \times T_Q=8 \times 0.2=1.6\ \mu s$$

代入式（9.2），可得

488

$$NBR=1/T_{BIT}=1/1.6\ \mu s=625\ 000\ B/s \text{ 或 } 625\ Kb/s$$

为了补偿总线上节点振荡器频率之间的相位差，每一个CAN控制器必须与接收信号的相关边沿保持同步。这里定义有两种类型的同步：硬件同步和再同步。硬件同步只在报文帧的开始时使用，此时每一个CAN节点必须将当前位时间的Sync_Seg同帧发送开始的显性或隐性边沿保持对准。按照同步规则，如果发生了硬件同步，那么在位时间内将不再发生再同步。

使用再同步，Phase_Seg1将被延长，或者Phase_Seg2将被缩短。在相位缓冲段内改变的数量是有一个上界的，这可以通过同步补偿宽度（SJW）来设定。SJW是可编程的，为1～4个不等，这个值需要加到Phase_Seg1上或者从Phase_Seg2中减去。

9.8 PIC微控制器CAN接口

通常，任何类型的PIC微控制器都可以应用于基于CAN总线的项目，但是一些PIC微控制器（如PIC18F258）带有内置的CAN模块，这能简化基于CAN总线的系统设计。不带有内置CAN总线模块的微控制器也可以在CAN总线应用中使用，但是需要另外添加硬件和软件，这使得设计更加昂贵和复杂。

489

图9-11给出了基于PIC微控制器的CAN总线应用方框图，这里使用的是不带内置CAN模块的PIC16或PIC12型的微控制器（如PIC16F84）。通过一个外部的MCP2515 CAN控制器芯片和一块MCP2551 CAN总线收发器芯片，将微控制器连接到CAN总线。这种配置很适合用来对一个使用任何PIC微控制器的现有设计进行更新升级。

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

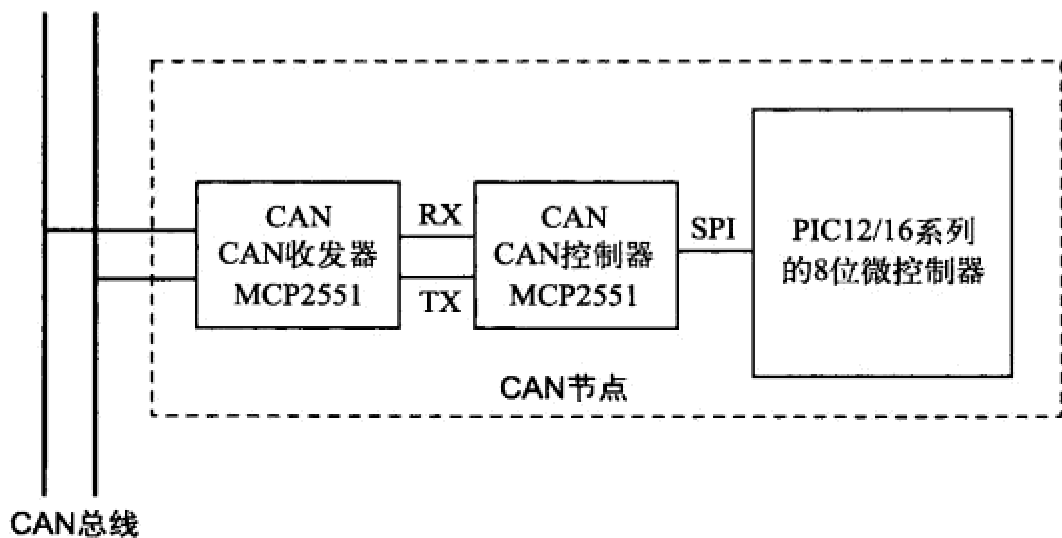


图9-11 使用PIC微控制器的CAN节点

对于新的基于CAN总线的设计，使用内置CAN模块的PIC微控制器更加容易。如图9-12所示，这种设备包括内置在芯片上的CAN控制器硬件。只需添加一个CAN收发器芯片就可设计一个CAN节点。表9-2列出了一些带有CAN模块的PIC微控制器。

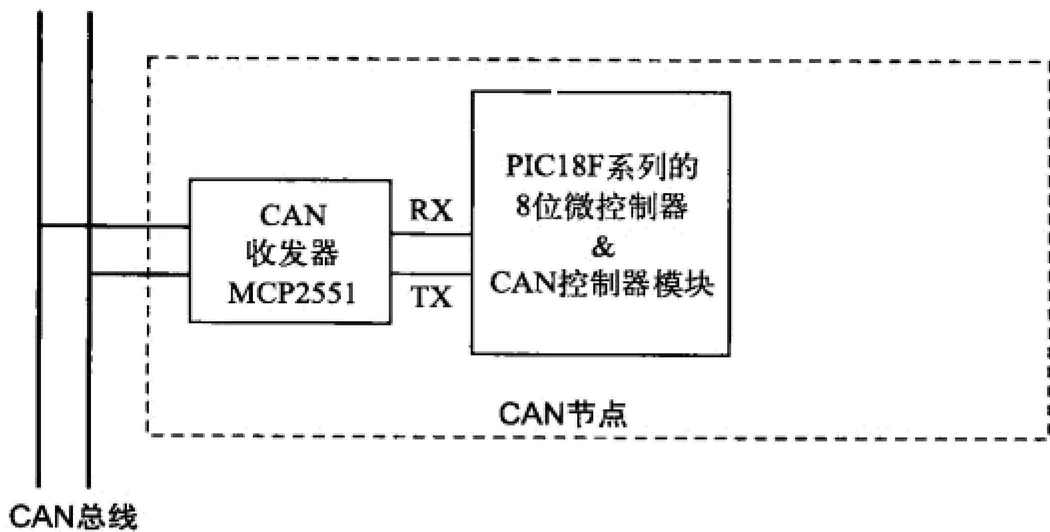


图9-12 使用集成CAN模块的CAN节点

表9-2 带有CAN模块的部分流行PIC微控制器

设备	引脚	闪存	SRAM	EEPROM	模数转换器	CAN模块	SPI	UART
18F258	28	16	768	256	5	1	1	1
18F2580	28	32	1536	256	8	1	1	1
18F2680	28	64	3328	1024	8	1	1	1
18F4480	40/44	16	768	256	11	1	1	1
18F8545	80	48	3328	1024	16	1	1	1
18F8680	80	64	3328	1024	16	1	1	1

490

9.9 PIC18F258 微控制器

本章稍后将在基于CAN总线的项目中使用PIC18F258微控制器。本节将介绍该微控制器及其有关内置CAN总线的工作原理。工作原理同样适用于其他带有CAN模块的PIC微控制器。

PIC18F258是一个集成了CAN模块的高性能8位微控制器。该设备具有如下的特点：

- 32 kbit闪存；

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

- 1 536 B RAM数据存储器；
- 256 B EEPROM存储器；
- 22个I/O端口；
- 5通道10位的A/D转换器；
- 3个定时器/计数器；
- 3个外部中断引脚；
- 大电流（25 mA）灌入/拉出能力；
- 捕捉/比较/PWM模块；
- SPI/I²C 模块；
- CAN2.0A/B模块；
- 上电复位和上电定时器；
- 看门狗定时器；
- 优先级中断；
- 直流电到40MHz交流电时钟输入；
- 8×8硬件乘法器；
- 宽范围的工作电压（2.0 V～5.5 V）；
- 省电睡眠模式。

491

PIC18F258微控制器的CAN模块具有如下的特性：

- 兼容CAN 1.2、CAN 2.0A和CAN 2.0B；
- 支持标准格式和扩展格式的数据帧；
- 可编程的位速率高达1 Mbit/s；
- 双缓冲器的接收器；
- 3个发送缓冲器；
- 2个接收缓冲器；
- 可编程的时钟源；
- 6个接收滤波器；
- 2个接收滤波屏蔽器；
- 用于自检的回环模式；
- 低功耗睡眠模式；
- 中断能力。

CAN模块使用端口引脚RB3/CANRX和RB2/CANTX，分别用于CAN总线接收和发送。这些引脚通过一个MCP2551型的CAN总线收发芯片连接到CAN总线。

PIC18F258微控制器支持如下的帧类型：

- 标准格式的数据帧；
- 扩展格式的数据帧；
- 遥控帧；
- 错误帧；
- 过载帧；
- 帧间间隔。

节点使用滤波器来判断是否接受一个接收到的报文。报文滤波被应用于整个识别码字段，而屏蔽寄存器则用来说明滤波器应该检查识别码中的哪一位。

492

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

PIC18F258微控制器的CAN模块有6个运行模式：

- 配置模式；
- 禁止模式；
- 正常工作模式；
- 监听模式；
- 回环模式；
- 错误识别模式。

9.9.1 配置模式

CAN模块在配置模式下进行初始化。当正在进行发送操作时，是不允许该模块进入配置模式的。在配置模式下，CAN模块既不能发送，也不能接收，错误计数器将被清零，中断标志保持不变。

9.9.2 禁止模式

在禁止模式下，CAN模块既不能发送，也不能接收。在该模式下，内部时钟停止工作，直到模块被激活。如果模块被激活，那么它将在总线上等待11个隐性位，犹如IDLE总线一样检测其状态，然后接受模块禁止命令。WAKIF中断（唤醒中断）是在禁止模式下唯一活动的CAN模块中断。

9.9.3 正常工作模式

正常工作模式是CAN模块的标准工作模式。在该模式下，模块监视所有的总线报文，并产生应答位、错误帧等。这是唯一能发送报文的模式。

9.9.4 监听模式

监听模式允许CAN模块接收报文，包括有错误的报文。它可以用来监视总线活动或者检测总线上的波特率。对于自动的波特率检测，要求至少有两个其他的节点正在进行通信。通过测试不同值直到接收到正确的报文，可以确定波特率。监听模式不能发送报文。 493

9.9.5 回环模式

在回环模式下，报文从外部发送缓冲器直接传送到接收缓冲器，而在总线上不进行实际的报文发送。在系统开发和测试期间，这种模式是很有用的。

9.9.6 错误识别模式

错误识别模式用来忽略所有的错误，并接收所有的报文。在该模式下，所有的报文，不管是有效的还是无效的，都被接收并复制到接收缓冲器中。

9.9.7 CAN 报文发送

PIC18F258微控制器提供了3个专用的发送缓冲器：TXB0、TXB1和TXB3。即将发送的报文按优先级队列排列。在发送SOF之前，对等待发送的所有缓冲器队列的优先级进行比较。具有最高优先级的发送缓冲器将最先被发送。如果两个缓冲器具有相同的优先级，那么具有较大缓冲器编号的先被发送。报文发送有4个优先级。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

9.9.8 CAN 报文接收

报文接收是一个更为复杂的过程。PIC18F258微控制器包括两个接收缓冲器RXB0和RXB1，每一个接收缓冲器都有多个接收滤波器（如图9-13所示）。所有接收到的报文都被集中到报文集成缓冲器（MAB）。一旦接收到一条报文，无论是识别码的类型还是数据字节的数量，整个报文都将被复制到MAB中。

接收到的报文具有优先级。RXB0是更高优先级的缓冲器，它有两个报文接收滤波器，即RXF0和RXF1。RXB1是较低优先级的缓冲器，有4个接收滤波器：RXF2、RXF3、RXF4和RXF5。两个可编程接收滤波屏蔽器（RXM0和RXM1），也是可用的，每个接收缓冲器对应着一个接收滤波屏蔽器。

494

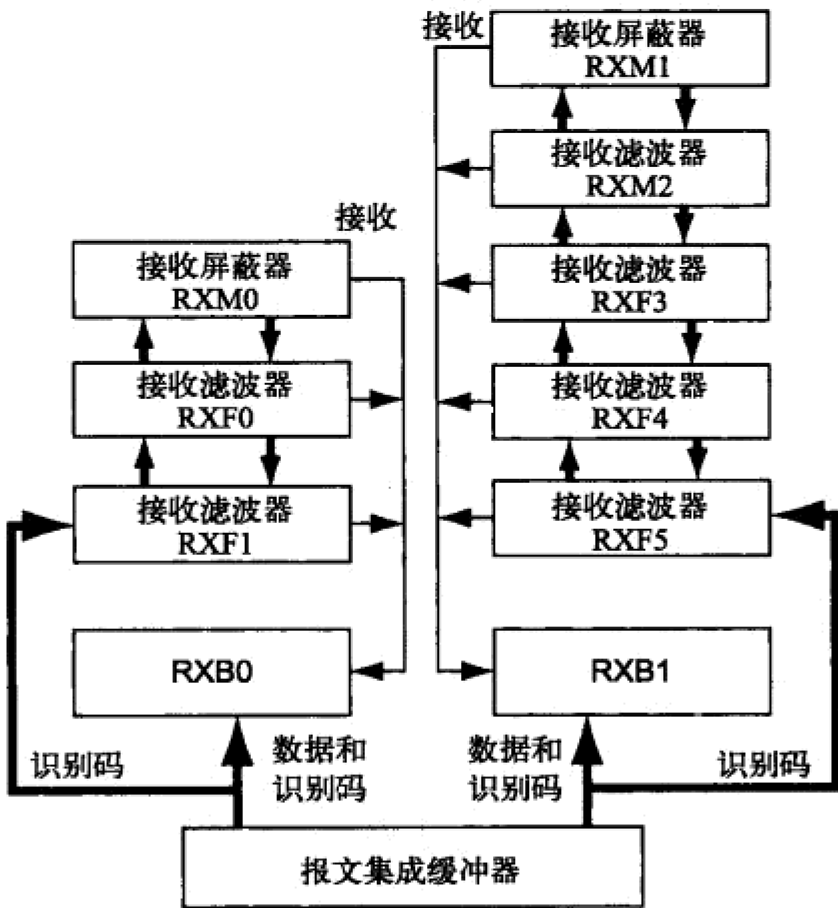


图9-13 接收缓冲器方框图

CAN模块使用报文接收滤波器和屏蔽器来决定MAB中的一条报文是否应该被加载到接收缓冲器。一旦MAB接收到一条有效的报文，就将该报文的识别码字段与滤波器的值进行比较。如果两者匹配，那么该报文就被加载到合适的接收缓冲器。滤波屏蔽器用来决定识别码中的哪一位将滤波器检查。表9-3的真值表说明了识别码中的每一位是如何与屏蔽器和滤波器的值进行比较的，进而确定报文是否应该被接受。如果屏蔽位被设置为0，那么不管滤波位如何，识别码中的对应位都将被自动接收。

495

表9-3 滤波器/屏蔽器真值表

屏蔽位	滤波位	报文识别码	接受或拒绝位
0	×	×	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

9.9.9 计算时序参数

节点的时序参数设置对于总线工作的可靠性非常重要。给定微控制器时钟频率和期望的CAN总线位速率，可以计算出以下的时序参数值：

- 波特率预分频器值
- Pro_Seg值
- Phase_Seg1值
- Phase_Seg2值
- SJW值

正确的时序要求满足：

- Pro_Seg + Phase_Seg1 ≥ Phase_Seg2
- Phase_Seg2 ≥ SJW

接下来的例子将说明这些时序参数的计算。

例9.2 假设微控制器振荡器时钟频率为20 MHz，期望的CAN位速率为125 kHz，请计算时序参数。

解 使用20MHz的时钟频率，则时钟周期为50 ns。选择波特率预分频器值为4，根据式(9.4)，T_Q=2*（BRP+1）*T_{OSC}，可以得到时间量子T_Q=500 ns。为了得到125 kHz的标称位速率，则标称的位时间必须为：

$$T_{BIT}=1/0.125\text{MHz}=8\mu\text{s}，或者16T_Q$$

496

Sync_segment为1T_Q。选择Prop_Seg为2T_Q，Phase_Seg1为7T_Q，Phase_Seg2为6T_Q，在Phase_Seg1末端设置采样点为10T_Q。

使用前面描述的规则，SJW是允许的最大值（即为4）。然而，只有在不同节点的时钟不稳定或不准确（例如使用陶瓷谐振器）时，才需要大的SJW。通常SJW为1就足够了。总之，所求的时序参数为：

```
波特率预分频器（BRP）  =4
Sync_seg              =1
Prop_Seg              =2
Phase_Seg1            =7
Phase_Seg2            =6
SJW                   =1
```

采样点位于10T_Q，对应于总的位时间的62.5%。

因特网上有一些免费的软件，用来计算CAN总线的时序参数。其中一个软件是CAN Baud Rate Calculator，这是由Artic咨询有限公司（http://www.articconsultants.co.uk）开发的。下面是使用这个软件的例子。

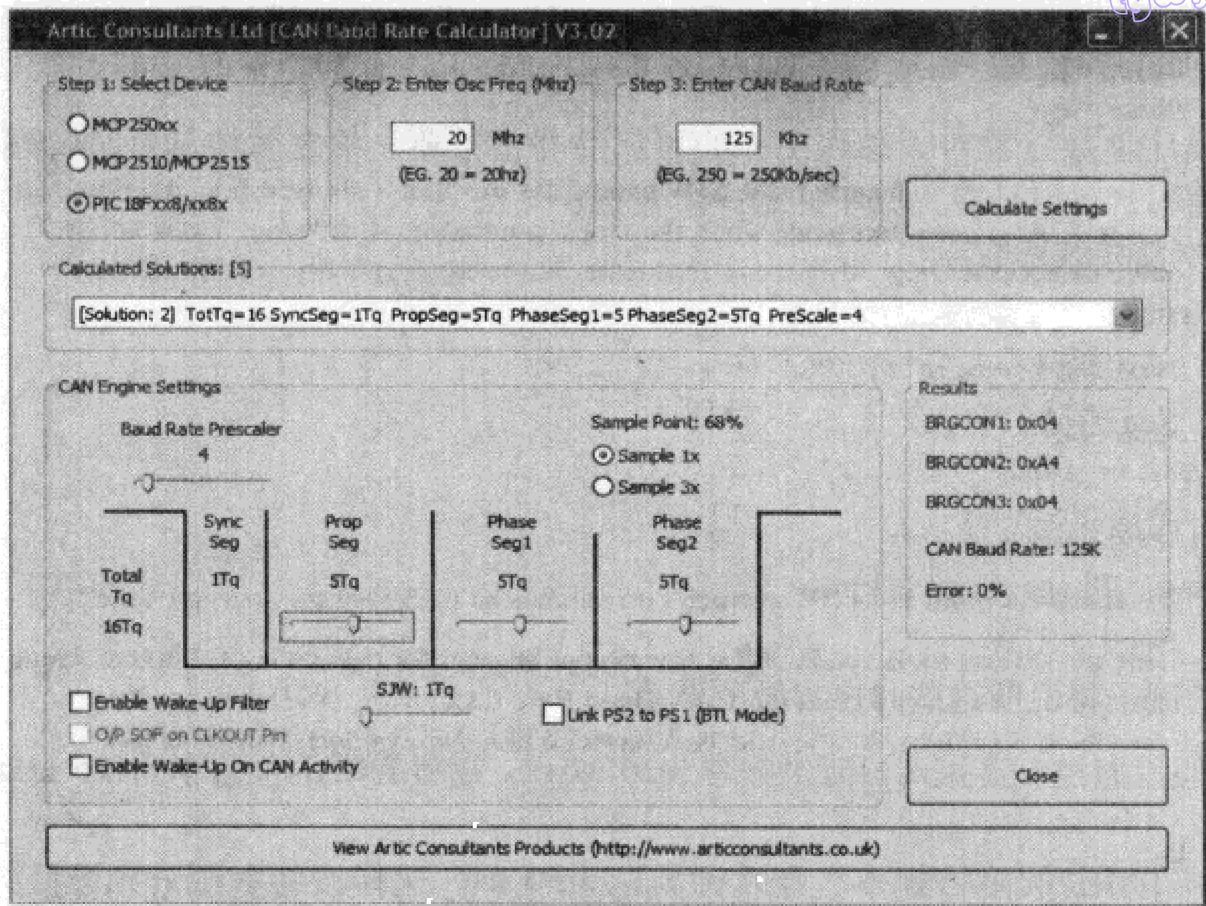
例9.3 假设微控制器振荡器时钟频率为20 MHz，期望的CAN位速率为125 kHz，使用CAN Baud Rate Calculator计算时序参数。

解 图9-14给出了CAN Baud Rate Calculator程序的输出结果。选择设备类型为PIC18xxx8，振荡器频率为20 MHz，CAN总线波特速率为125 kHz。

单击Calculate Settings按钮，计算和显示推荐的时序参数。总的来说，这里可用的方法不止一种，可以在Calculate Solutions字段的下拉菜单中选择不同的方法。

世纪电源网·论坛
BBS.21dianyuan.com
电源工程师
设计灵感之源

tyw藏书



497

图9-14　CAN Baud Rate Calculator程序的输出

从下拉菜单中选择方法2，使用程序计算得到的推荐时序参数为：

```
波特率预分频器（BRP）  =4
Sync_seg              =1
Prop_Seg              =5
Phase_Seg1            =5
Phase_Seg2            =5
SJW                   =1
采样点                 =68%
错误                   =0%
```

9.10　mikroC CAN函数

498

对于CAN总线应用，mikroC语言提供了两个库：用于带有内置CAN模块的PIC微控制器的库和对于不带内置CAN模块的PIC微控制器且使用SPI总线的库。本节只讨论带有内置CAN模块的PIC微控制器可用的函数库。类似的函数也可用于不带内置CAN模块的PIC微控制器。

mikroC CAN函数仅被使用MCP2551或者类似的CAN收发器的PIC18XXX8微控制器支持。标准（11个识别码位）和扩展（29个识别码位）格式的报文均被支持。

提供的mikroC函数如下：

- CANSetOperationMode
- CANGetOperationMode
- CANInitialize
- CANSetBaudRate
- CANSetMask
- CANSetFilter
- CANRead
- CANWrite



世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

9.10.1 CANSetOperationMode

CANSetOperationMode函数用来设置CAN工作模式。该函数原型为：

```
void CANSetOperationMode(char mode,char wait_flag)
```

参数wait_flag为0或0xFF。如果被设置为0xFF，则函数阻滞且不会返回任何值，直到设置为所需的模式。如果被设置为0，则函数以一个无阻塞调用返回。

CAN模式可以是下列模式之一：

- CAN_MODE_NORMAL　　正常工作模式
- CAN_MODE_SLEEP　　睡眠工作模式
- CAN_MODE_LOOP　　回环工作模式
- CAN_MODE_LISTEN　　监听工作模式
- CAN_MODE_CONFIG　　配置工作模式

499

9.10.2 CANGetOperationMode

CANGetOperationMode函数用来返回当前的CAN工作模式。该函数原型为：

```
char CANGetoperationMode(void)
```

9.10.3 CANIntialize

CANInitialize函数用来初始化CAN模块。将所有屏蔽寄存器清零，以允许所有的报文。在执行函数时，设置为正常模式。该函数原型为：

```
void CANInitialize(char sjw,char BRP,char PHSEG1,char PHSEG2,
char PROPEG,char CAN_CONFIG_FLAGS)
```

其中，

SJW	是同步补偿宽度
BRP	是波特率预分频器
PHSEG1	是Phase_Seg1时序参数
PHSEG1	是Phase_Seg1时序参数
PROPSEG	是Prop_Seg

CAN_CONFIG_FLAGS可以是下述的配置标志之一：

- CAN_CONFIG_DEFAULT　　默认标志
- CAN_CONFIG_PHASEG2_PRG_ON　　使用提供的PHSEG2值
- CAN_CONFIG_PHASEG2_PRG_OFF　　使用PHSEG1的最大值或者信息处理时间，无论哪个更大
- CAN_CONFIG_LINE_FILTER_ON　　对于唤醒，使用CAN总线滤波器
- CAN_CONFIG_FILTER_OFF　　不使用CAN总线滤波器
- CAN_CONFIG_SAMPLE_ONCE　　在采样点采样总线一次
- CAN_CONFIG_SAMPLE_THRICE　　在采样点之前采样总线三次
- CAN_CONFIG_STD_MSG　　只接受标准格式的识别码报文
- CAN_CONFIG_XTD_MSG　　只接受扩展格式的识别码报文
- CAN_CONFIG_DBL_BUFFER_ON　　使用双缓冲器来接收数据
- CAN_CONFIG_DBL_BUFFER_OFF　　不使用双缓冲器
- CAN_CONFIG_ALL_MSG　　接受所有的报文，包括无效的

500

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

kuw藏书

- CAN_CONFIG_VALID_XTD_MSG 只接受有效的扩展格式识别码报文
- CAN_CONFIG_VALID_STD_MSG 只接受有效的标准格式识别码报文
- CAN_CONFIG_ALL_VALID_MSG 接受所有的有效报文

这些配置值可以进行逐位的逻辑AND操作，以形成复杂的配置值。

9.10.4 CANSetBaudRate

CANSetBaudRate函数用来设置CAN总线的波特率。该函数原型为：

```
void CANSetBaudRate(char SJW, char BRP,char PHSEG1,char PHSEG2,
char PROPSEG,char CAN_CONFIG_FLAGS)
```

函数的形参与函数CANInitialize类似。

9.10.5 CANSetMask

501

CANSetMask函数用来为滤波器报文设置屏蔽器。该函数原型为：

```
void CANSetMask(char CAN_MASK,long Value,Char
CAN_CONFIGFLAGS)
```

CAN_MASK可以是下述情况之一：

- CAN_MASK_B1 接收缓冲器1屏蔽器值
- CAN_MASK_B2 接收缓冲器2屏蔽器值

变量value为屏蔽寄存器值。CAN_CONFIG_FLAGS可以是CAN_CONFIG_XTD（扩展格式的报文）或者是CAN_CONFIG_STD（标准格式的报文）。

9.10.6 CANSetFilter

CANSerFilter函数用来设置滤波器值。该函数原型为：

```
void_CANSetFilter(char CAN_FILTER,long value,char
CAN_CONFIG_FLAGS)
```

CAN_FILTER可以是下述情形之一：

- CAN_FILTER_B1_F1 用于缓冲器1的滤波器1
- CAN_FILTER_B1_F2 用于缓冲器2的滤波器1
- CAN_FILTER_B2_F1 用于缓冲器1的滤波器2
- CAN_FILTER_B2_F2 用于缓冲器2的滤波器2
- CAN_FILTER_B2_F3 用于缓冲器3的滤波器2
- CAN_FILTER_B2_F4 用于缓冲器4的滤波器2

CAN_CONFIG_FLAGS 可以是 CAN_COONFIG_XTD（扩展格式的报文），或者是 CAN_COONFIG_STD（扩展格式的报文）。

9.10.7 CANRead

CANRead函数用来从CAN总线读取报文。如果没有有效的报文，则返回0。该函数原型为：

502

```
char CANRead(long*id, char*data, char*datalen, char
* CAN_RX_MSG_FLAGS)
```

变量id是CAN报文的识别码。只有11或29位可供使用，这取决于报文的类型（标准格式或者扩展格式）。变量data是最大长度为8 B的数组，用来存储接收到的数据。变量datalen是接收到的数据长度（1～8）。

CAN_RX_MSG_FLAGS可以是下述情况之一：

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

lyw藏书

- CAN_RX_FILTER_1　　接收缓冲滤波器1已接受该报文
- CAN_RX_FILTER_2　　接收缓冲滤波器2已接受该报文
- CAN_RX_FILTER_3　　接收缓冲滤波器3已接受该报文
- CAN_RX_FILTER_4　　接收缓冲滤波器4已接受该报文
- CAN_RX_FILTER_5　　接收缓冲滤波器5已接受该报文
- CAN_RX_FILTER_6　　接收缓冲滤波器6已接受该报文
- CAN_RX_OVERFLOW　　已发生接收缓冲器溢出
- CAN_RX_INVALID_MSG　　已接收的无效报文
- CAN_RX_XTD_FRAME　　已接收到扩展格式的识别码报文
- CAN_RX_RTR_FRAME　　已接收到RTR帧报文
- CAN_RX_DBL_BUFFERED　　该报文使用双缓冲器

如果需要，可以将这些标志进行逐位的逻辑AND运算。

9.10.8 CANWrite

CANWrite函数用来向CAN总线发送报文。如果报文不能排队（缓冲器满），则返回0。该函数原型为：

```
char CANWrite(long id,char*data,char datalen,char
CAN_TX_MSG_FLAGS)
```

变量id是CAN报文的识别码。只有11或29位可供使用，这取决于报文的类型（标准格式或者扩展格式）。变量data是长度最大为8 B的数组，用来存储要发送的数据。变量datalen是数据的长度（1～8）。

CAN_RX_MSG_FLAGS可以是下述情形之一：

- CAN_TX_PRIORITY_0　　发送优先级 0
- CAN_TX_PRIORITY_1　　发送优先级 1
- CAN_TX_PRIORITY_2　　发送优先级 2
- CAN_TX_PRIORITY_3　　发送优先级 3
- CAN_TX_STD_FRAME　　标准格式的识别码报文
- CAN_TX_XTD_FRAME　　扩展格式的识别码报文
- CAN_TX_NO_RTR_FRAME　　非RTR报文
- CAN_TX_RTR_FRAME　　RTR报文

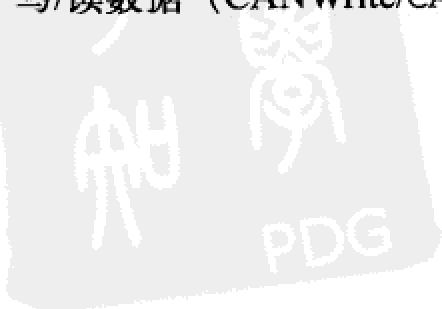
如果需要，可以将这些标志进行逐位的逻辑AND运算。

503

9.11 CAN 总线编程

为了在CAN总线上运行PIC18F258微控制器，需要执行以下的步骤：

- 配置CAN总线的I/O端口方向（RB2和RB3）；
- 初始化CAN模块（CANInitialize）；
- 设置CAN模块为CONFIG模式（CANSetOperationMode）；
- 设置屏蔽寄存器（CANSetMask）；
- 设置滤波器寄存器（CANSetFilter）；
- 设置CAN模块为正常模式（CANSetOperationMode）；
- 写/读数据（CANWrite/CANRead）。



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

项目9.1 温度传感器CAN总线项目

下面是一个简单的基于CAN总线的双节点项目。图9-15给出该项目的方框图。该系统由两个节点组成。其中一个节点（叫作DISPLAY节点）负责每秒请求一次温度值，并显示在LCD上。这个过程将连续地重复。另一个节点（叫作COLLECTOR节点）负责从外部的半导体温度传感器读取温度值。

504

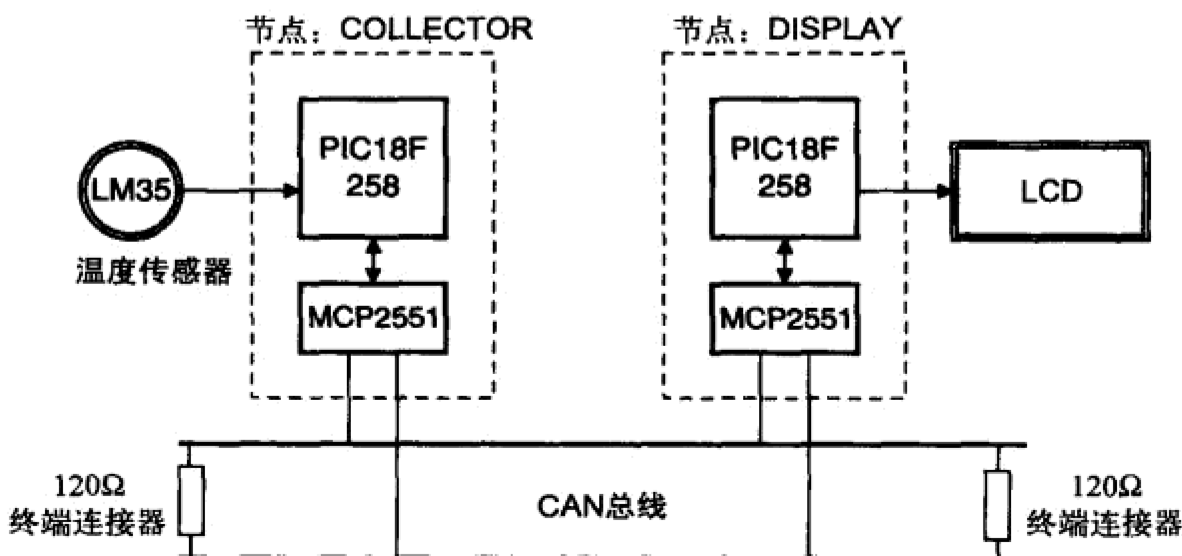
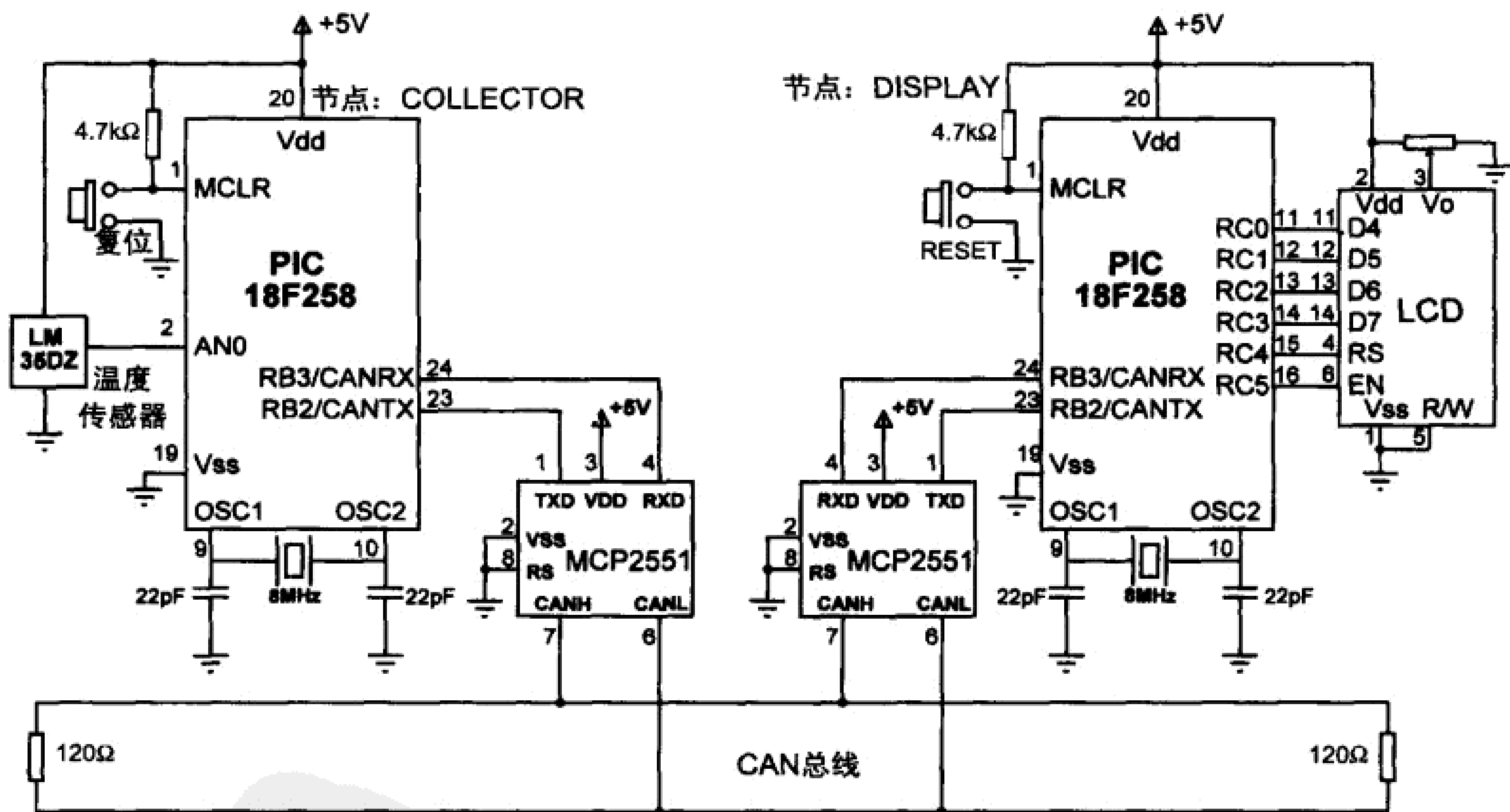


图9-15 项目的方框图

项目的电路原理图如图9-16所示。两个CAN节点使用一条2 m长的双绞线连接在一起，在CAN总线的每一个末端使用一个120 Ω的电阻来终止。



505

图9-16 项目的电路图

DISPLAY处理器

类似于COLLECTOR处理器，DISPLAY处理器由一个内置CAN模块的PIC18F258微控制器和一个MCP2551收发器芯片组成。微控制器工作在8MHz的晶体振荡频率下。MCLR输入被连接到一个外部的复位按钮。微控制器的CAN输出引脚（RB2/CANTX和RB3/CANRX）被连接到MCP2551的输入引脚TXD和RXD。HD44780型LCD被连接到微控制器的PORTC，用来显示温度值。

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

COLLECTOR处理器

COLLECTOR处理器包括一个内置CAN模块的PIC18F258微控制器和一个MCP2551收发器芯片。微控制器工作在8 MHz的晶振频率下。MCLR输入连接到一个外部的复位按钮。微控制器的模拟输入引脚AN0连接到一个LM35DZ型的半导体温度传感器。该传感器能测量0 ℃～100 ℃范围内的温度，并产生一个与测量温度成正比的模拟电压（即输出为10 mV/℃）。例如，在20 ℃时，输出电压为200 mV。

微控制器的CAN输出引脚（RB2/CANTX和RB3/CANRX）被连接到MCP2551型CAN收发器芯片的输入引脚TXD和RXD。该芯片的CANH和CANL输出引脚被直接通过一条双绞线连接到CAN总线。MCP2551是一个8引脚的芯片，它支持的数据速率达到1 Mb/s。这个芯片能驱动112个节点。在该芯片的引脚8处连接一个外部电阻，用以控制CANH和CANL的上升沿时间和下降沿时间，以降低EMI。对于高速操作的情况，该引脚应该被接地。值为VDD/2的参考电压是从芯片的引脚5输出的。

该项目的程序清单分为两个部分：DISPLAY程序和COLLECTOR程序。该系统的工作如下。

506

- DISPLAY处理器通过CAN总线向COLLECTOR处理器请求当前的温度值。
- COLLECTOR处理器读取温度值，并进行格式化，然后通过CAN总线发送给DISPLAY处理器。
- DISPLAY处理器从CAN总线读取温度值，然后将其显示在LCD上。
- 该过程每隔1秒钟重复一次。

DISPLAY程序

图9-17给出了DISPLAY程序（即DISPLAY.C）的程序清单。在程序的开始处，将PORTC设置为输出，将RB3设置为输入（CANRX），将RB2设置为输出（CANTX）。在这个项目中，选择CAN总线的位速率为100 Kb/s。微控制器的时钟频率为8 MHz，使用Baud Rate Calculation程序（如图9-14所示）来计算时序参数：

```
SJW=1
BRP=1
Phase_Seg1=6
Phase_Seg2=7
Prop_Seg=6
```

```
/**************************************************************************
                CAN BUS EXAMPLE - NODE: DISPLAY
                ==============================

This is the DISPLAY node of the CAN bus example. In this project a PIC18F258
type microcontroller is used. An MCP2551 type CAN bus transceiver is used to
connect the microcontroller to the CAN bus. The microcontroller is operated from
an 8MHz crystal with an external reset button.

Pin CANRX and CANTX of the microcontroller are connected to pins RXD
and TXD of the transceiver chip respectively. Pins CANH and CANL of
the transceiver chip are connected to the CAN bus.

An LCD is connected to PORTC of the microcontroller. The ambient
temperature is read from another CAN node and is displayed on the LCD.

The LCD is connected to the microcontroller as follows:

Microcontroller     LCD

  RC0             D4
  RC1             D5
  RC2             D6
  RC3             D7
  RC4             RS
  RC5             EN
```

图9-17 DISPLAY程序清单

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
CAN speed parameters are:

    Microcontroller clock:      8MHz
    CAN Bus bit rate:           100Kb/s
    Sync_Seg:                   1
    Prop_Seg:                   6
    Phase_Seg1:                 6
    Phase_Seg2:                 7
    SJW:                        1
    BRP:                        1
    Sample point:               65%

Author:     Dogan Ibrahim
Date:       October 2007
File:       DISPLAY.C
******************************************************************************/

void main()
{
    unsigned char temperature, data[8];
    unsigned short init_flag, send_flag, dt, len, read_flag;
    char SJW, BRP, Phase_Seg1, Phase_Seg2, Prop_Seg, txt[4];
    long id, mask;

     TRISC = 0;                          // PORTC are outputs (LCD)
     TRISB = 0x08;                       // RB2 is output, RB3 is input
 //
 // CAN BUS Parameters
 //
     SJW = 1;
     BRP = 1;
     Phase_Seg1 = 6;
     Phase_Seg2 = 7;
     Prop_Seg = 6;

     init_flag = CAN_CONFIG_SAMPLE_THRICE  &
             CAN_CONFIG_PHSEG2_PRG_ON  &
             CAN_CONFIG_STD_MSG        &
             CAN_CONFIG_DBL_BUFFER_ON  &
             CAN_CONFIG_VALID_XTD_MSG  &
             CAN_CONFIG_LINE_FILTER_OFF;

     send_flag = CAN_TX_PRIORITY_0        &
             CAN_TX_XTD_FRAME         &
             CAN_TX_NO_RTR_FRAME;

     read_flag = 0;
 //
 // Initialize CAN module
 //
      CANInitialize(SJW, BRP, Phase_Seg1, Phase_Seg2, Prop_Seg, init_flag);
 //
 // Set CAN CONFIG mode
 //
      CANSetOperationMode(CAN_MODE_CONFIG, 0xFF);

      mask = -1;
 //
 // Set all MASK1 bits to 1's
 //
      CANSetMask(CAN_MASK_B1, mask, CAN_CONFIG_XTD_MSG);
```

图9-17　（续）

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
//
// Set all MASK2 bits to 1's
//
    CANSetMask(CAN_MASK_B2, mask, CAN_CONFIG_XTD_MSG);
//
// Set id of filter B2_F3 to 3
//
    CANSetFilter(CAN_FILTER_B2_F3,3,CAN_CONFIG_XTD_MSG);
//
// Set CAN module to NORMAL mode
//
    CANSetOperationMode(CAN_MODE_NORMAL, 0xFF);
//
// Configure LCD
//
   Lcd_Config(&PORTC,4,5,0,3,2,1,0);       // LCD is connected to PORTC
   Lcd_Cmd(LCD_CLEAR);                     // Clear LCD
   Lcd_Out(1,1,"CAN BUS");                 // Display heading on LCD
   Delay_ms(1000);                         // Wait for 2 seconds

//
// Program loop. Read the temperature from Node:COLLECTOR and display
// on the LCD continuously
//
   for(;;)                                 // Endless loop
   {
     Lcd_Cmd(LCD_CLEAR);                   // Clear LCD
     Lcd_Out(1,1,"Temp = ");               // Display "Temp = "
     //
     // Send a message to Node:COLLECTOR and ask for data
     //
     data[0] = 'T';                        // Data to be sent
     id = 500;                             // Identifier
     CANWrite(id, data, 1, send_flag);     // send 'T'
     //
     // Get temperature from node:COLLECT
     //
     dt = 0;
     while(!dt)dt = CANRead(&id, data, &len, &read_flag);
     if(id == 3)
     {
       temperature = data[0];
       ByteToStr(temperature,txt);         // Convert to string
       Lcd_Out(1,8,txt);                   // Output to LCD
       Delay_ms(1000);                     // Wait 1 second
     }
   }

}
```

图9-17 （续）

mikroC CAN总线函数CANInitialize是用来初始化CAN模块的。时序参数和初始化标志被指定为函数中的形参。初始化标志由逐位的逻辑AND运算来完成：

```
init_flag=CAN_CONFIG_SAMPLE_THRICE &
          CAN_CONFIG_PHSEG2_RPG_ON &
          CAN_CONFIG_STD_MSG &
          CAN_CONFIG_DBL_BUFFER_ON &
          CAN_CONFIG_VALID_XTD_MSG &
          CAN_CONFIG_LINE_FILTER_OFF;
```

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

这里指定了采样总线3次，指定了标准格式的识别码，打开了双缓冲器，且关闭了总线滤波器。

然后，将工作模式设置为配置模式（CONFIG），指定滤波屏蔽器和滤波器值。将屏蔽寄存器1和屏蔽寄存器2的值都设为全1(−1是十六进制数FFFFFFFF的速记方式，即把所有屏蔽位都置为1)，以使所有的滤波器位都能与收到的数据匹配。

507

将用于缓冲器2的滤波器3设置为数值3，以使值为3的识别码都能被接收缓冲器接受。

接下来，将工作模式设置为正常模式（NORMAL）。然后，程序设置LCD，并在LCD上显示报文“CAN BUS”约1秒钟。

508～510

主程序连续地循环执行，并以for语句开始。在循环体中，LCD被清屏，在LCD上显示文本信息“TEMP=”。接下来，使用识别码500，将字符“T”通过总线进行发送（COLLECTOR节点滤波器被设置为接受识别码500）。这是一个发送到COLLECTOR节点的报文，用以请求温度读数。然后，程序从CAN总线上读取温度，将其转化为用数组txt存储的字符串，并显示在LCD上。在延迟1秒钟后，重复上述过程。

COLLECTOR程序

图9-18给出了COLLECTOR程序（即COLLECTOR.C）的程序清单。该程序的初始化部分与DISPLAY程序相同。将接收滤波器设置为500，这样使用识别码500的报文都可以被程序接受。

```
/*****************************************************************************
                    CAN BUS EXAMPLE - NODE: COLLECTOR
                    ==================================

This is the COLLECTOR node of the CAN bus example. In this project a
PIC18F258 type microcontroller is used. An MCP2551 type CAN bus transceiver
is used to connect the microcontroller to the CAN bus. The microcontroller is
operated from an 8MHz crystal with an external reset button.

Pin CANRX and CANTX of the microcontroller are connected to pins RXD
and TXD of the transceiver chip respectively. Pins CANH and CANL of the
transceiver chip are connected to the CAN bus.

An LM35DZ type analog temperature sensor is connected to port AN0 of the
microcontroller. The microcontroller reads the temperature when a request is
received and then sends the temperature value as a byte to Node:DISPLAY on
the CAN bus.

CAN speed parameters are:

    Microcontroller clock:      8MHz
    CAN Bus bit rate:           100Kb/s
    Sync_Seg:                   1
    Prop_Seg:                   6
    Phase_Seg1:                 6
    Phase_Seg2:                 7
    SJW:                        1
    BRP:                        1
    Sample point:               65%

Author:         Dogan Ibrahim
Date:           October 2007
File:           COLLECTOR.C
*****************************************************************************/

void main()
{
    unsigned char temperature, data[8];
    unsigned short init_flag, send_flag, dt, len, read_flag;
    char SJW, BRP, Phase_Seg1, Phase_Seg2, Prop_Seg, txt[4];
```

图9-18 COLLECTOR程序清单

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
    unsigned int temp;
    unsigned long mV;
    long id, mask;

    TRISA = 0xFF;                    // PORTA are inputs
    TRISB = 0x08;                    // RB2 is output, RB3 is input
//
// Configure A/D converter
//
    ADCON1 = 0x80;

//
// CAN BUS Timing Parameters
//
    SJW = 1;
    BRP = 1;
    Phase_Seg1 = 6;
    Phase_Seg2 = 7;
    BRP = 1;
    Prop_Seg = 6;

    init_flag = CAN_CONFIG_SAMPLE_THRICE &
            CAN_CONFIG_PHSEG2_PRG_ON &
            CAN_CONFIG_STD_MSG       &
            CAN_CONFIG_DBL_BUFFER_ON &
            CAN_CONFIG_VALID_XTD_MSG &
            CAN_CONFIG_LINE_FILTER_OFF;

    send_flag = CAN_TX_PRIORITY_0       &
            CAN_TX_XTD_FRAME         &
            CAN_TX_NO_RTR_FRAME;

    read_flag = 0;
//
// Initialise CAN module
//
    CANInitialize(SJW, BRP, Phase_Seg1, Phase_Seg2, Prop_Seg, init_flag);
//
// Set CAN CONFIG mode
//
    CANSetOperationMode(CAN_MODE_CONFIG, 0xFF);

    mask = -1;
//
// Set all MASK1 bits to 1's
//
    CANSetMask(CAN_MASK_B1, mask, CAN_CONFIG_XTD_MSG);
//
// Set all MASK2 bits to 1's
//
    CANSetMask(CAN_MASK_B2, mask, CAN_CONFIG_XTD_MSG);
//
// Set id of filter B1_F1 to 3
//
    CANSetFilter(CAN_FILTER_B2_F3,500,CAN_CONFIG_XTD_MSG);
//
// Set CAN module to NORMAL mode
//
    CANSetOperationMode(CAN_MODE_NORMAL, 0xFF);

//
```

图9-18 （续）

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

lyuu藏书

```
// Program loop. Read the temperature from analog temperature
// sensor
//
    for(;;)                                          // Endless loop
    {
      //
      // Wait until a request is received
      //
      dt = 0;
      while(!dt) dt = CANRead(&id, data, &len, &read_flag);
      if(id == 500 && data[0] == 'T')
      {
          //
          // Now read the temperature
          //
          temp = Adc_Read(0);                        // Read temp
          mV = (unsigned long)temp * 5000 / 1024;    // in mV
          temperature = mV/10;                       // in degrees C
          //
          // send the temperature to Node:Display
          //
          data[0] = temperature;
          id = 3;                                    // Identifier
          CANWrite(id, data, 1, send_flag);          // send temperature
      }
    }
}
```

图9-18 （续）

在程序循环体内，程序一直等待，直到接收到发送温度的请求。在这里，请求通过接受字符“T”来进行识别。一旦接收到有效的请求，温度就被读取并转换为℃（存储于变量temperature），然后与识别码值3作为一个字节被发送到CAN总线。这个过程将永远地重复。

图9-19对两个节点的工作做了小结。

Node: DISPLAY

Initialize CAN module
Set mode to CONFIG
Set Mask bits to 1's
Set Filter value to 3
Set mode to NORMAL

DO FOREVER
    Send character "T" with *identifier* 500
    Read temperature with *identifier* 3
    Convert temperature to string
    Display temperature on LCD
    Wait 1 second
ENDDO

Node: COLLECTOR

Initialize CAN module
Set mode to CONFIG
Set Mask bits to 1's
Set Filter value to 500
Set mode to NORMAL

DO FOREVER
    Read a character
    IF character is "T"
        Read temperature
        Convert to digital
        Convert to °C
        Send with *identifier* 3
    ENDIF
ENDDO

图9-19 两个节点的工作

511～514

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

第 10 章　多任务和实时操作系统

几乎所有基于微控制器的系统都执行一个以上的任务。例如，一个温度监控系统由3个任务组成，通常它们在一个短暂的延迟后又会重复，即：

- 任务1读取温度值
- 任务2格式化温度值
- 任务3显示温度值

更复杂的系统可能有许多复杂的任务。在一个多任务系统中，众多的任务要求CPU时间，因为只有一个CPU，所以需要某种形式的组织和协调来给每个任务分配它需要的CPU时间。实际上，每个任务只能取得一个非常简短的时间，因此看起来所有的任务都是并行地同时执行的。

几乎所有基于微控制器的系统都在实时地工作。实时系统是一个时间应答系统，它能在最短的时间内对外部环境作出应答。实时并非一定意味着微控制器应该以高速运行。在一个实时系统中，最重要的是快速的应答时间，尽管高速度对实时更有帮助。例如，假设有一个基于微控制器的实时系统，连接有几个外部开关，当一个开关发生动作或者某个别的事件发生时，该系统应该能作出立即的应答。

一个实时操作系统（RTOS）是指一段程序代码（通常称为内核），当微控制器工作在多任务环境下时，它用来控制任务的分配。例如，RTOS可以决定下一个要执行的任务，可以决定如何协调任务的优先级，以及决定如何在任务之间传送数据和报文。

515

本章将探讨多任务嵌入式系统的基本原理，并给出在简单项目中应用RTOS的例子。多任务代码和RTOS是复杂而宽泛的课题，本章只是简单地描述有关这些工具的概念。有兴趣的读者，请参阅关于操作系统、多任务系统和RTOS的更多书籍和论文。

对于PIC 微控制器，有几个商用的RTOS系统。在本书编写之时，mikroC语言不提供内置的RTOS。用于PIC微控制器的两个流行的高级RTOS系统是Salvo（www.pumkin.com）（可以使用高科技的PIC C编译器）和CCS（Customer Computer Services）内置RTOS系统。在本章中，RTOS项目例子是基于CCS（www.ccsinfo.com）编译器的，这是一个专门为PIC16和PIC18系列微控制器开发的PIC C编译器。

10.1　状态机

状态机是一个简单的结构，通常用来执行以序列形式出现的几个任务。现实生活中的许多系统都属于这个范畴。例如，使用状态机结构，可以更容易地描述洗衣机或者洗碗机的操作。

或许，在C语言中，实现状态机结构的最简单方法是使用switch-case语句。例如，我们的温度监视系统有3个任务，分别被命名为任务1、任务2和任务3，如图10-1所示。图10-2给出了使用switch-case语句实现3个任务的状态机。初始状态是任务1，将每个任务的状态号增加1，来选择下一个要执行的状态。最后一个状态选择状态1，在switch-case语句的末尾有一个时间延迟。状态机结构在一个无限的for循环内被连续地执行。

516

PDG

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

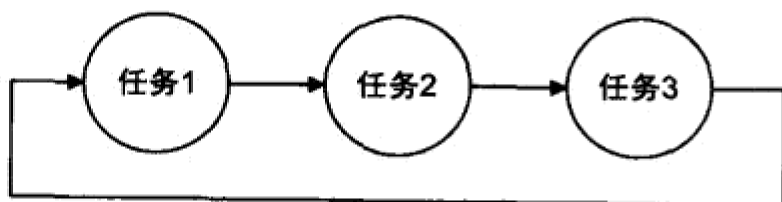


图10-1　状态机的实现

```
for(;;)
{
        state = 1;
        switch (state)
        {
                CASE 1:
                        implement TASK 1
                        state++;
                        break;
                CASE 2:
                        implement TASK 2
                        state++;
                        break;
                CASE 3:
                        implement TASK 3
                        state = 1;
                        break;
        }
        Delay_ms(n);
}
```

图10-2　状态机的C语言实现

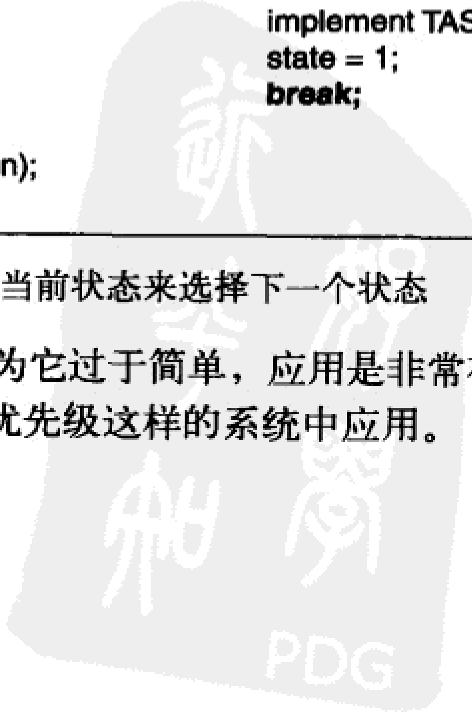
在许多的应用中，状态不需要按序列执行。相反地，当前状态直接地或基于某些条件来选择下一个状态。如图10-3所示。

```
for(;;)
{
        state = 1;
        switch (state)
        {
                CASE 1:
                        implement TASK 1
                        state = 2;
                        break;
                CASE 2:
                        implement TASK 2
                        state = 3;
                        break;
                CASE 3:
                        implement TASK 3
                        state = 1;
                        break;
        }
        Delay_ms(n);
}
```

517

图10-3　从当前状态来选择下一个状态

尽管状态机很容易实现，但是正因为它过于简单，应用是非常有限的。它们只能在无需准确应答、任务活动有严格定义、任务无优先级这样的系统中应用。



世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

而且，一些任务可能比其他的任务更重要。有些任务在适当的时候必须运行。例如，在一个制造工厂，当温度太高时，必须执行用来开启警报器的任务。这种任务的实现需要一个复杂的系统（如RTOS）。

10.2 实时操作系统（RTOS）

实时操作系统是围绕多任务内核来建立的，该内核用来控制任务的时间片分配。时间片是指一个已知任务在被其他任务停止或者代替之前的执行时间。这个过程将被连续地重复，也叫作内容切换。当内容切换发生时，正在执行的任务被停止，处理器寄存器被存储到存储器中，下一个可用任务的处理器寄存器被加载到CPU，新的任务开始执行。RTOS也提供任务对任务的消息传递、任务同步和共享资源的分配。

一个RTOS的基本组成部分包括：

- 调度器
- RTOS服务
- 同步和消息传递工具

调度器

调度器位于每个RTOS的中心，因为它提供算法来选择要执行的任务。三个常见的调度算法是：

- 协作调度
- 轮询调度
- 抢先调度

518

协作调度或许是最简单的调度算法。每一个任务运行，直到完成并主动放弃CPU。协作调度不能满足实时系统的需要，因为它不能按照重要性来确定任务的优先级。此外，一个单独的任务可能使用CPU很长时间，只留下少许的时间给其他的任务。调度器不能控制不同任务的执行时间。状态机结构就是协作调度技术的一个简单形式。

在轮询调度中，每个任务都被分配相等的CPU时间，如图10-4所示。一个计数器用来跟踪每个任务的时间片。当一个任务的时间片完成时，计数器清零，并将该任务置于循环的末端。新增的任务被置于循环的末端，并且将计数器清零。同协作调度一样，这在实时系统中并不是很有用，因为通常一些任务只占用几毫秒，而其他的任务则需要几百毫秒，甚至更多的时间。

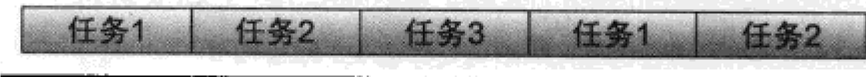


图10-4 轮询调度

抢先调度被认为是一个实时调度算法。它是基于优先级的，每个任务被赋予一个优先级，如图10-5所示。具有最高优先级的任务将得到CPU时间。实时系统通常支持范围从0～255的优先级，其中0为最高优先级，255为最低优先级。

519

在一些实时系统中，存在多个任务可能处于相同的优先级的情况，因此可以混合使用抢先调度和轮询调度。在这种情形中，处于较高优先级的任务将在较低级的任务之前运行，处于相同优先级的任务通过轮询调度来运行。如果一个任务被一个更高优先级的任务抢先，那么它的运行时间计数器将被保存，在重新得到CPU的控制时将被恢复。

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

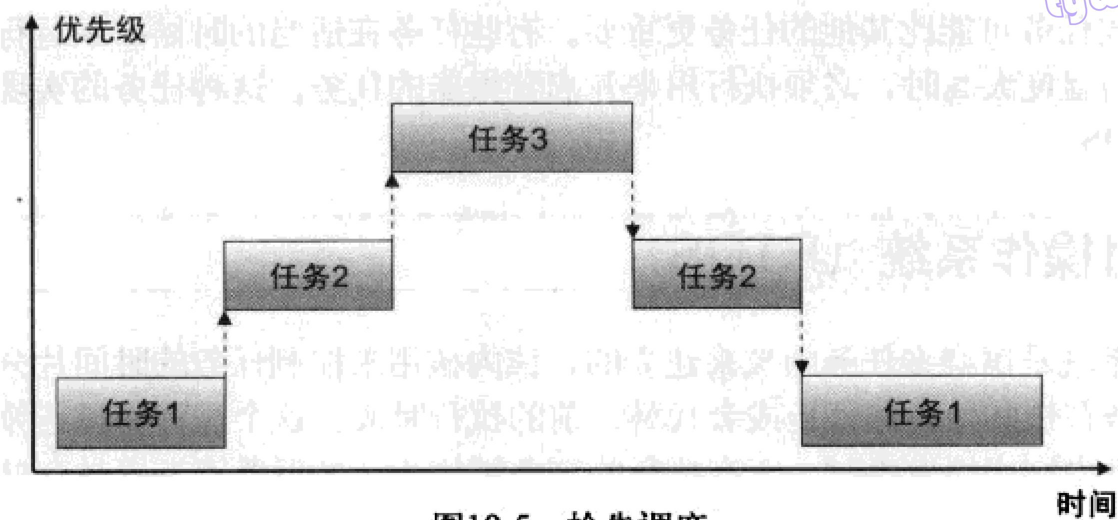


图10-5　抢先调度

在一些系统中，定义了严格的实时优先级，在该优先级以上的任务可以运行到完成（或者一直运行，直到资源不可用），即使有其他的任务处于相同的优先级。

在一个实时系统中，任务可以处于下述的状态之一（如图10-6所示）：

- 准备运行
- 运行中
- 阻断

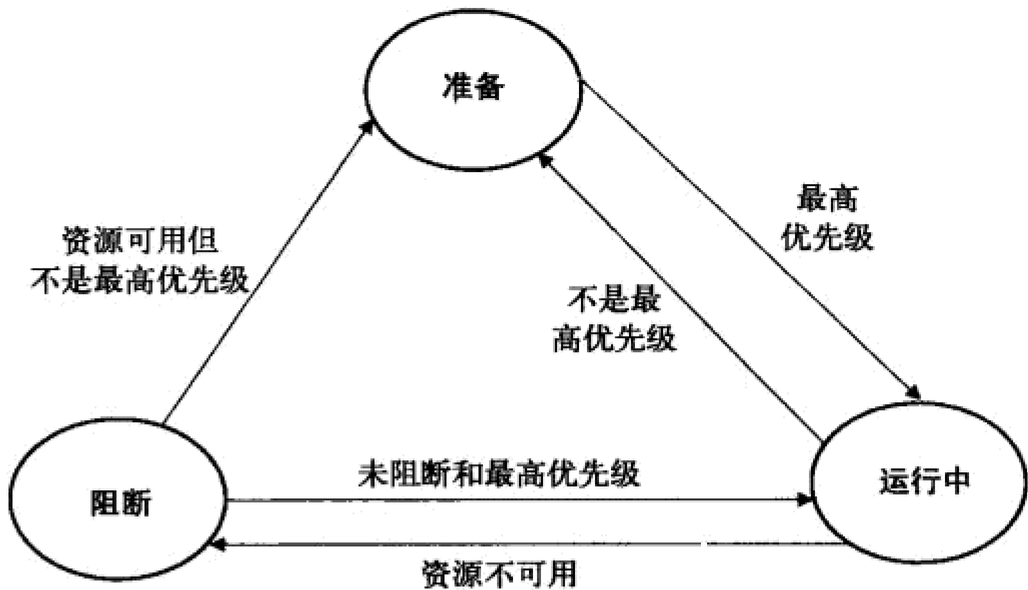


图10-6　任务状态

当任务第一次被创建时，它的状态通常是准备运行，并进入到任务列表。从这个状态起，遵循调度算法，该任务可以变为运行中的任务。按照抢先调度的条件，如果该任务成为系统中最高优先级的任务，且不需要等待资源，那么该任务将执行。

如果需要的资源不可用，那么运行中的任务也会变成一个阻断的任务。例如，一个任务需要从A/D转换器读取数据，它将会被阻断，直到数据可用。一旦需要的资源可访问，如果阻断的任务成为系统中最高优先级的任务，那么该任务将成为运行中的任务。否则，它将进入准备状态。只有运行中的任务才会被阻断。一个准备状态的任务是不会被阻断的。

520

当任务从一个状态进入另一个状态时，处理器将在存储器中保存运行中任务的内容，并从存储器加载新任务的内容，然后执行期望的新指令。

内核通常提供一个接口来操纵任务的运行。常用的任务操作有：

- 创建任务
- 删除任务

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

tyw藏书

■ 改变任务的优先级
■ 改变任务的状态

10.3 RTOS 服务

RTOS服务是内核提供的有用工具，用来帮助开发人员高效地创建实时任务。例如，一个任务可以使用时间服务来获得当前的数据和时间。这些服务中可以包括：

■ 中断处理服务
■ 时间服务
■ 设备管理服务
■ 存储器管理服务
■ 输入输出服务

10.4 同步和消息工具

同步和消息工具是核心结构，用来帮助开发人员创建实时应用。这些服务包括：

521

■ 信号量
■ 事件标志
■ 邮箱
■ 管道
■ 消息队列

信号量被用来同步访问共享资源，如公共的数据区域。时间标志被用来同步任务之间的活动。邮箱、管道和消息队列被用来在任务之间发送消息。

10.5 CCS PIC C 编译器 RTOS

CCS PIC C编译器是一种流行的C编译器，专门为PIC16和PIC18系列的微控制器而设计。除了PIC编译器之外，CCS（Customer Computer Services）公司提供PIC内电路模拟器、仿真器、微控制器编程器和各种开发套件。CCS C语言的语法与mikroC语言略有不同，但是熟悉mikroC的读者将会发现CCS C很容易使用。

对于使用PCW和PCWH编译器的PIC18系列微控制器，CCS C支持基础的多任务协作RTOS。该RTOS允许PIC微控制器执行任务时不使用中断。当调度执行一个任务时，该任务将被赋予处理器的控制权。当一个任务执行完毕或者不再需要处理器时，控制将返回到调度函数，该调度函数将把处理器的控制交给下一个被调度的任务。因为RTOS不使用中断而且不是抢先式，用户必须保证一个任务不会永远地运行。关于RTOS的更多技术细节，请参阅编译器的用户手册。

除标准的C函数之外，CCS语言还提供下述的RTOS函数。

函数rtos_run()用来初始化RTOS的运行。所有任务的控制操作都在调用该函数之后才被执行。

函数rtos_terminate()用来结束RTOS的运行。控制返回到没有RTOS的原始程序。事实上，该函数如同一个从函数rtos_run()的返回。

函数rtos_enable()接收一个任务的名字作为形参。该函数用来使能一个任务，因此，当

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

tyw藏书

522 时间到期时，函数rtos_run()可以调用这个任务。

函数rtos_disable()接收一个任务的名字作为形参。该函数用来禁止一个任务，因此，该任务将不再被函数rtos_run()调用，除非通过调用函数rtos_enable()再次使能该任务。

函数rtos_yield()，当在一个任务内调用该函数时，控制将返回到调度程序。所有的任务都应该调用该函数来释放处理器，这样其他的任务才能使用处理器时间。

函数rtos_msg_send()接收一个任务的名字和一个字节数据作为形参。该函数发送这个字节数据到指定的任务，该字节数据位于任务的报文队列中。

函数rtos_msg_read()用来读取位于任务报文队列中的字节数据。

函数rtos_msg_poll()返回真值，如果任务报文队列里有数据。该函数应该在从任务报文队列读取一个字节数据之前被调用。

函数rtos_signal()接收一个信号量名字，并将信号量进行增量。

函数rtos_wait()接收一个信号量名字，并等待与信号量有关的资源变为可用。然后，递减信号量，以使任务能请求资源。

函数rtos_await()接收一个表达式作为形参，该任务将一直等待，直到表达式的值为真。

函数rtos_overrun()接收一个任务名字作为形参，如果任务超过了为它分配的时间，该函数将返回真值。

函数rtos_stats()返回关于指定任务的指定统计数据。该统计数据可以是最小的任务运行次数、最大的任务运行次数和总的任务运行次数。任务的名字和统计数据的类型被指定为该函数的形参。

10.5.1 准备使用 RTOS

除了前面介绍的函数以外，在调用任何RTOS函数之前，必须在函数的开头指定预处理器命令#use rtos()。该预处理器命令的格式为：

```
#use rtos(timer=n,minor_cycle=m)
```

其中，变量timer的取值为0～4，用来说明RTOS将使用的处理器定时器，变量minor_cycle

523 是每个任务运行的最长时间。必须在这里输入的数后紧跟s、ms、μs或者ns。

另外，在minor_cyle选项之后，还可以指定statistics选项。在这种情况下，编译器将跟踪每次调用任务所使用的最小处理器时间和最大处理器时间，以及任务使用的总时间。

10.5.2 声明任务

任务的声明跟任何其他的C函数一样，但是在多任务应用中的任务是没有参数的，也不返回任何值。在声明任务之前，需要使用预处理器命令#task来说明任务选项。该预处理器命令的格式为：

```
#task(rate=n,max=m,queue=p)
```

其中，rate用来指定任务应该被调用的频率。必须在所指定的数字后紧跟s、ms、μs或者ns。max用来指定在任务的一次执行期间任务将使用的处理器时间。这里所指定的时间必须等于或者小于minor_cycle指定的时间。queue是可选的，现在可以为任务指定要保留的字节数，用来从其他的任务接收报文。其默认值为0。

在下面的例子中，调用任务my_tick的周期为20 ms，要求使用的处理器时间不大于100 ms。该任务不使用queue选项，如：

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
#task(rate=20ms,max=100ms)
void my_ticks()
{
        ...........
        ...........
}
```

项目 10.1 LED(发光二极管)

在接下来的基于RTOS的简单项目里，4个LED被连接到一个PIC18F452型微控制器的PORTB的下半部分。软件由4个任务组成，其中每个任务以不同的速率闪烁一个LED。

- 任务1，称为task_B0，以每250ms一次的速率让连接到端口RB0的LED闪烁。
- 任务2，称为task_B1，以每500ms一次的速率让连接到端口RB1的LED闪烁。
- 任务3，称为task_B2，以每秒一次的速率让连接到端口RB2的LED闪烁。
- 任务4，称为task_B3，以每两秒一次的速率让连接到端口RB3的LED闪烁。

524

图10-7给出了该项目的电路图。这里使用一个4MHz的晶体振荡器作为时钟。PORTB的引脚RB0～RB3通过限流电阻被连接到LED。

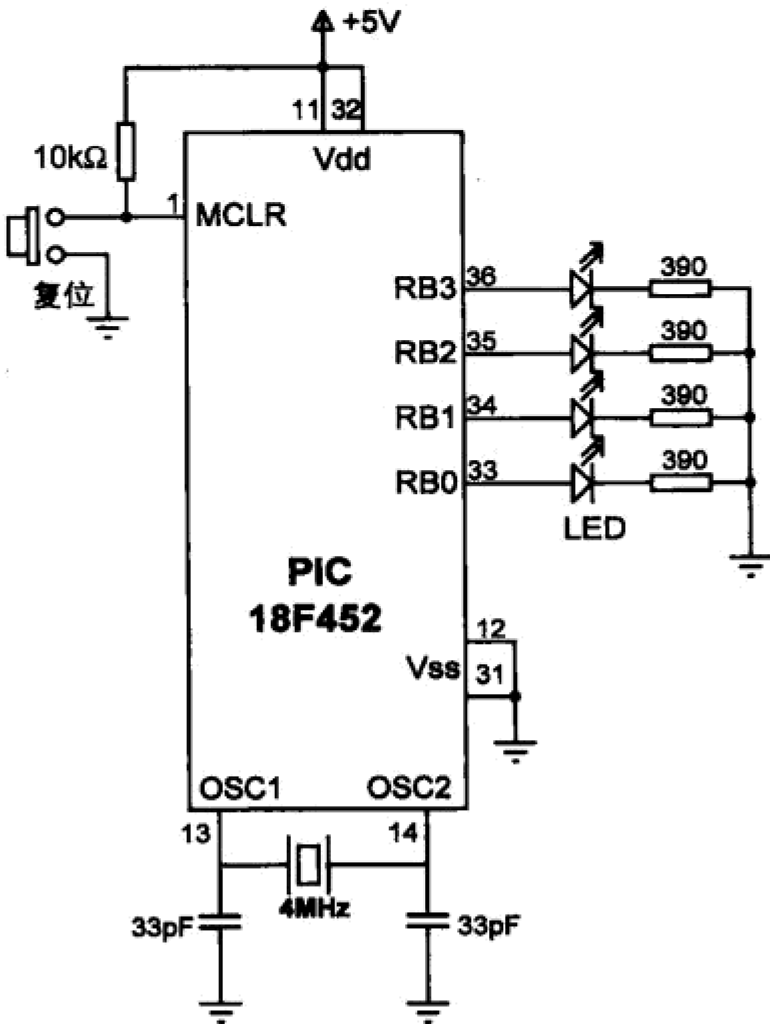


图10-7 项目的电路图

软件以CCS C编译器为基础，其程序清单（RTOS1.C）如图10-8所示。主程序位于程序的末尾，在主程序的内部，PORTB引脚被声明为输出，RTOS通过调用函数rtos_run()来开始。

```
//////////////////////////////////////////////////////////////////////////////////////////
//
//                    SIMPLE RTOS EXAMPLE
//                    ----------------------------------
//
```

图10-8 项目的程序清单

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
// This is a simple RTOS example. 4 LEDs are connected to lower half of
// PORTB of a PIC18F452 microcontroller. The program consists of 4
// tasks:
//
// Task task_B0 flashes the LED connected to port RB0 every 250ms.
// Task task_B1 flashes the LED connected to port RB1 every 500ms.
// Task task_B2 flashes the LED connected to port RB2 every second
// Task task_B3 flashes the LED connected to port RB3 every 2 seconds.
//
// The microcontroller is operated from a 4MHz crystal
//
// Programmer:      Dogan Ibrahim
// Date:            September, 2007
// File:            RTOS1.C
//
//////////////////////////////////////////////////////////////////////////////////////////
#include "C:\NEWNES\PROGRAMS\rtos.h"
#use delay (clock=4000000)

//
// Define which timer to use and minor_cycle for RTOS
//
  #use rtos(timer=0, minor_cycle=10ms)

//
// Declare TASK 1 - called every 250ms
//
  #task(rate=250ms, max=10ms)
  void task_B0()
  {
    output_toggle(PIN_B0);                          // Toggle RB0
  }


//
// Declare TASK 2 - called every 500ms
//
  #task(rate=500ms, max=10ms)
  void task_B1()
  {
    output_toggle(PIN_B1);                          // Toggle RB1
  }

//
// Declare TASK 3 - called every second
//
  #task(rate=1s, max=10ms)
  void task_B2()
  {
    output_toggle(PIN_B2);                          // Toggle RB2
  }

//
// Declare TASK 4 - called every 2 seconds
//
  #task(rate=2s, max=10ms)
  void task_B3()
  {
    output_toggle(PIN_B3);                          // Toggle RB3
  }
```

图10-8 （续）

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
//
// Start of MAIN program
//
void main()
{
  set_tris_b(0);                          // Configure PORTB as outputs
  rtos_run();                             // Start RTOS
}
```

图10-8 （续）

包含CCS RTOS声明的文件应该被包括在程序的开始处。预处理器命令#use delay将告诉编译器使用的是4 MHz时钟。然后，使用预处理器命令#use rtos，将RTOS定时器声明为Timer 0，将minor_cycle时间声明为10 ms。

525～526

该程序由4个相似的任务组成。

- task_B0使连接到RB0的LED以250 ms的速率闪烁。于是，LED打开250 ms，然后关闭250 ms，如此循环。在每次调用任务时，CCS状态output_toggle被用来改变LED状态。在CCS编译器中，PIN_B0是指微控制器的端口引脚RB0。
- task_B1使连接到RB1的LED以每500 ms一次的速率闪烁。
- task_B2使连接到RB2的LED以每秒一次的速率闪烁。
- task_B3使连接到RB3的LED以每两秒一次的速率闪烁。

527

图10-8所示的程序是一个多任务的程序，其中LED可以相互独立地同时闪烁。

项目 10.2 随机数发生器

在这个稍微复杂的RTOS项目里，将产生一个0～255之间的随机数。8个LED被连接到PIC18F452微控制器的PORTB。此外，一个按键开关被连接到PORTD（RD0）的位0，一个LED被连接到PORTD（RD7）的位7。

该项目使用了3个任务：Live、Generator和Display。

- 任务Live每隔200ms运行一次，用来点亮端口引脚RD7上的LED，以表明系统正在工作。
- 任务Generator用来连续地将变量从0增加到255，并检测按键开关的状态。当按键开关被压下时，使用一个报文队列将当前的计数值发送到任务Display。
- 任务 Display从报文队列读取数字，并将接收到的字节发送到连接在PORTB的LED。这样，每当按键开关被压下时，LED就会显示一个随机模式。

528

图10-9给出了该项目的方框图。图10-10给出了该项目的电路图。微控制器使用4MHz的晶体频率。

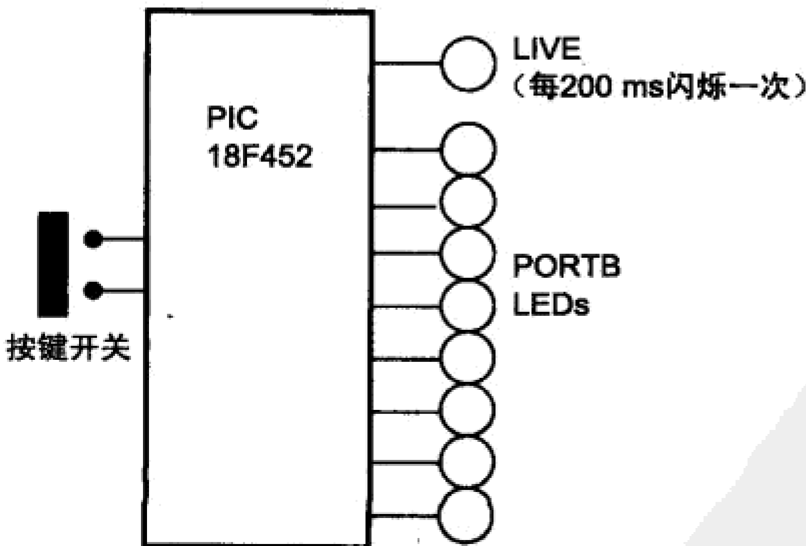


图10-9 项目的方框图

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

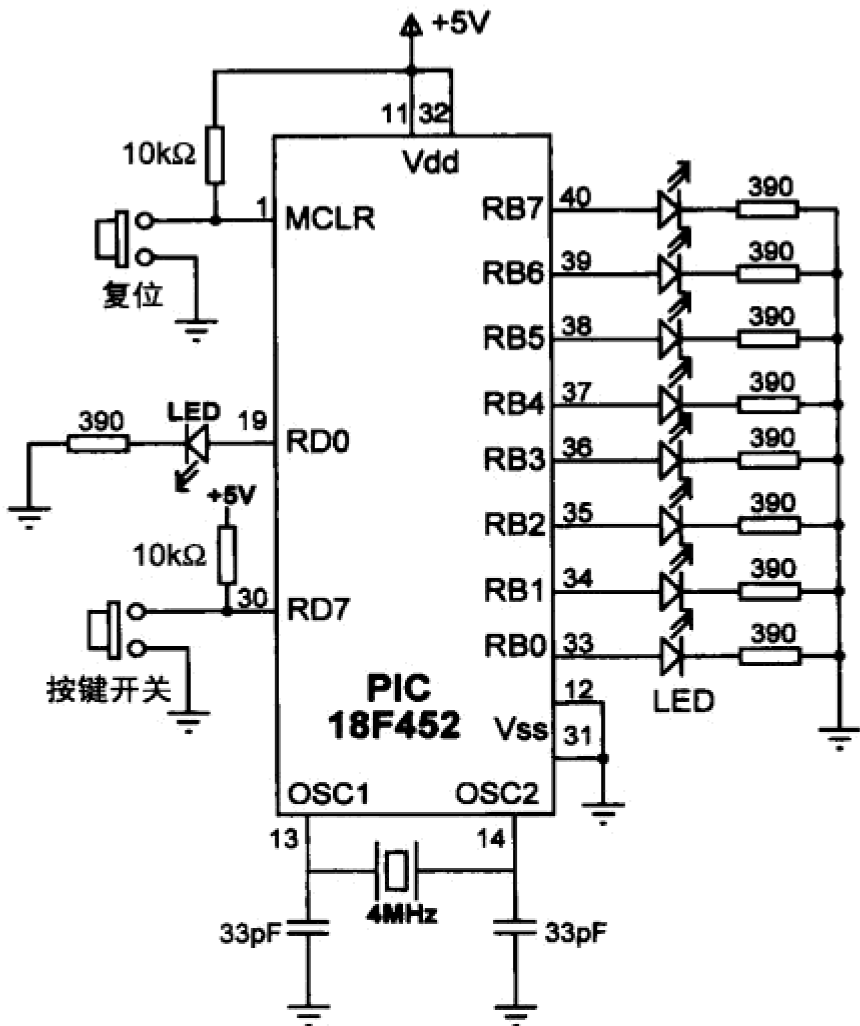


图10-10　项目的电路图

图10-11给出了该项目的程序清单（RTOS.C）。程序的主要部分位于程序的后半部分，它将PORTD引脚配置为输出。同样地，将PORTB的位0配置为输入，将PORTB的其他引脚配置为输出。

```
//////////////////////////////////////////////////////////////////////////////////////////////
//
//          SIMPLE RTOS EXAMPLE - RANDOM NUMBER GENERATOR
//          -----------------------------------------------------------------------------------
//
// This is a simple RTOS example. 8 LEDs are connected to PORTB
// of a PIC18F452 microcontroller. Also, a push-button switch is
// connected to port RC0 of PORTC, and an LED is connected to port
// RC7 of the microcontroller. The push-button switch is normally at logic 1.
//
// The program consists of 3 tasks called "Generator", "Display", and "Live".
//
// Task "Generator" runs in a loop and increments a counter from 0 to 255.
// This task also checks the state of the push-button switch. When the
// push-button switch is pressed, the task sends the value of the count to the
// "Display" task using messaging passing mechanism. The "Display" task
// receives the value of count and displays it on the PORTB LEDs.
//
// Task "Live" flashes the LED connected to port RC7 at a rate of 250ms.
// This task is used to indicate that the system is working and is ready for
// the user to press the push-button.
//
// The microcontroller is operated from a 4MHz crystal
//
// Programmer:        Dogan Ibrahim
// Date:              September, 2007
```

图10-11　项目的程序清单

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
// File:                 RTOS2.C
//
//////////////////////////////////////////////////////////////////////////
#include "C:\NEWNES\PROGRAMS\rtos.h"
#use delay (clock=4000000)
int count;

//
// Define which timer to use and minor_cycle for RTOS
//
  #use rtos(timer=0, minor_cycle=1ms)

//
// Declare TASK "Live" - called every 200ms
// This task flashes the LED on port RC7
//
  #task(rate=200ms, max=1ms)
  void Live()
  {
    output_toggle(PIN_D7);
  }


//
// Declare TASK "Display" - called every 10ms
//
  #task(rate=10ms, max=1ms, queue=1)
  void Display()
  {
    if(rtos_msg_poll() > 0)                    // Is there a message ?
    {
      output_b(rtos_msg_read());               // Send to PORTB
    }
  }


//
// Declare TASK "Generator" - called every millisecond
//
  #task(rate=1ms, max=1ms)
  void Generator()
  {
    count++;                                   // Increment count
    if(input(PIN_D0) == 0)                     // Switch pressed ?
    {
      rtos_msg_send(Display,count);            // send a message
    }
  }


//
// Start of MAIN program
//
void main()
{
  set_tris_b(0);                          // Configure PORTB as outputs
  set_tris_d(1);                          // RD0=input, RD7=output
  rtos_run();                             // Start RTOS
}
```

图10-11 （续）

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源

定时器0被用作RTOS的定时器，minor_cycle被设置为1 s。该程序包含3个任务。

■ 任务Live每隔200 ms运行一次，并使连接到端口引脚RD7的LED闪亮。该LED表示系统正在工作。
■ 任务Generator每隔1ms运行一次，并连续地增加字节变量count。当按键开关被压下时，PORTD的引脚0（RD0）将变为逻辑0。当发生这种情况时，使用RTOS函数调用rtos_msg_send(display,count)，将count的当前值发送到任务Display，其中Display是消息被发送到的任务名字，count是要发送的字节。
■ 任务Display每10 ms运行一次。该任务检查在队列里是否有报文。如果有，则使用RTOS函数调用rtos_msg_read()来提取报文，所读取到的字节将被发送到连接至PORTB的LED。因此，当开关被压下时，LED显示count的二进制值。应该使用函数rtos_msg_poll()来检测报文队列，因为在队列中没有任何字节时任何读取队列的尝试都会导致程序停止。

529
530～531

项目 10.3　使用 RS232 串行输出的电压表

这个RTOS项目比前面的项目都要复杂得多，在该项目中，使用A/D转换器读取电压值，然后通过串行端口将电压值发送到PC。该项目包括3个任务：Live、Get_voltage和To_RS232。

■ 任务Live每隔200ms运行一次，并使连接到微控制器端口RD7的LED闪烁，以表明系统正在工作。
■ 任务Get_voltage用来读取A/D转换器的通道0，这里假设被测量的电压已连接到A/D转换器。所读得的电压值在被格式化后，被储存在一个变量中。该任务每2 s运行一次。
■ 任务TO_RS232负责读取格式化后的电压值，并以每秒一次的速率通过RS232线向PC发送电压值。

图10-12给出了该项目的方框图。图10-13给出了该项目的电路图。这里使用PIC18F8520型微控制器(尽管可以使用任意的PIC18F系列微控制器），时钟频率由10 MHz的晶体振荡器提供。将待测量的电压连接到微控制器的模拟端口AN0。微控制器的RS232 TX输出引脚（RC6）被连接到一个MAN232型的RS232转换器芯片上，然后通过一个9引脚的D型连接器连接到PC的串行端口（即COM1）。将端口引脚RD7连接到一个LED，用以显示该项目是否正在工作。

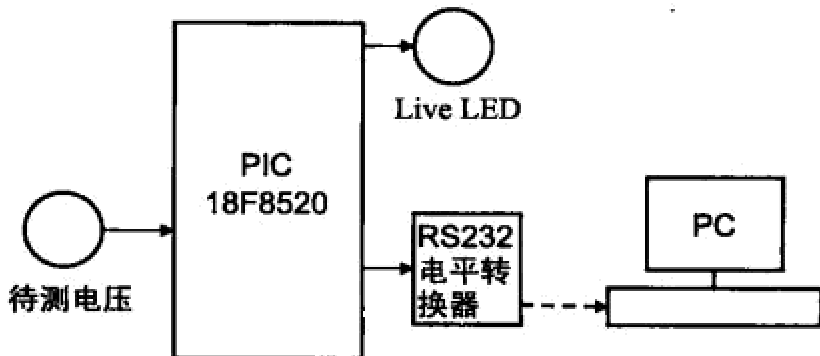


图10-12　项目的方框图

图10-14给出了该项目的程序清单（RTOS3.C）。在程序的开始处，将A/D定义为10位，将时钟定义为10 MHz，将RSR232的波特率定义为2 400。然后，使用预处理器命令#use rtos来定义RTOS定时器和minor_cycle。

532

在程序的主要部分，PORTD被配置为输出，所有的PORTD引脚被清零。然后，PORTA被配置为输入(RA0为模拟输入），微控制器的模拟输入被配置，A/D时钟被配置，A/D通道0（AN0）被选中。然后，调用函数rtos_run()，开始RTOS。

该程序包含3个任务。

■ 任务Live每200 ms运行一次，并使连接到微控制器端口引脚RD7的LED闪烁，以表明系统正在工作。
■ 任务Get_voltage从微控制器的通道0（引脚RA0或AN0）读取模拟电压。通过乘以5 000和除以1 024，将读取的电压值转换成毫伏（在一个10位的A/D中，有1 024个量化单位，当工作在+5 V的基准电压时，每个量化单位相当于5 000/1 024 mV）。该电压值被储存在一个叫作volts的全局变量中。

533

世纪电源网·论坛 电源工程师 设计灵感之源 BBS.21dianyuan.com

■ 任务To_RS232从公共变量Volts中读取已测量的电压值，并使用C printf语句将该电压值发送到RS232端口。该测量结果以如下的格式发送：

测量电压 = nnnn mV

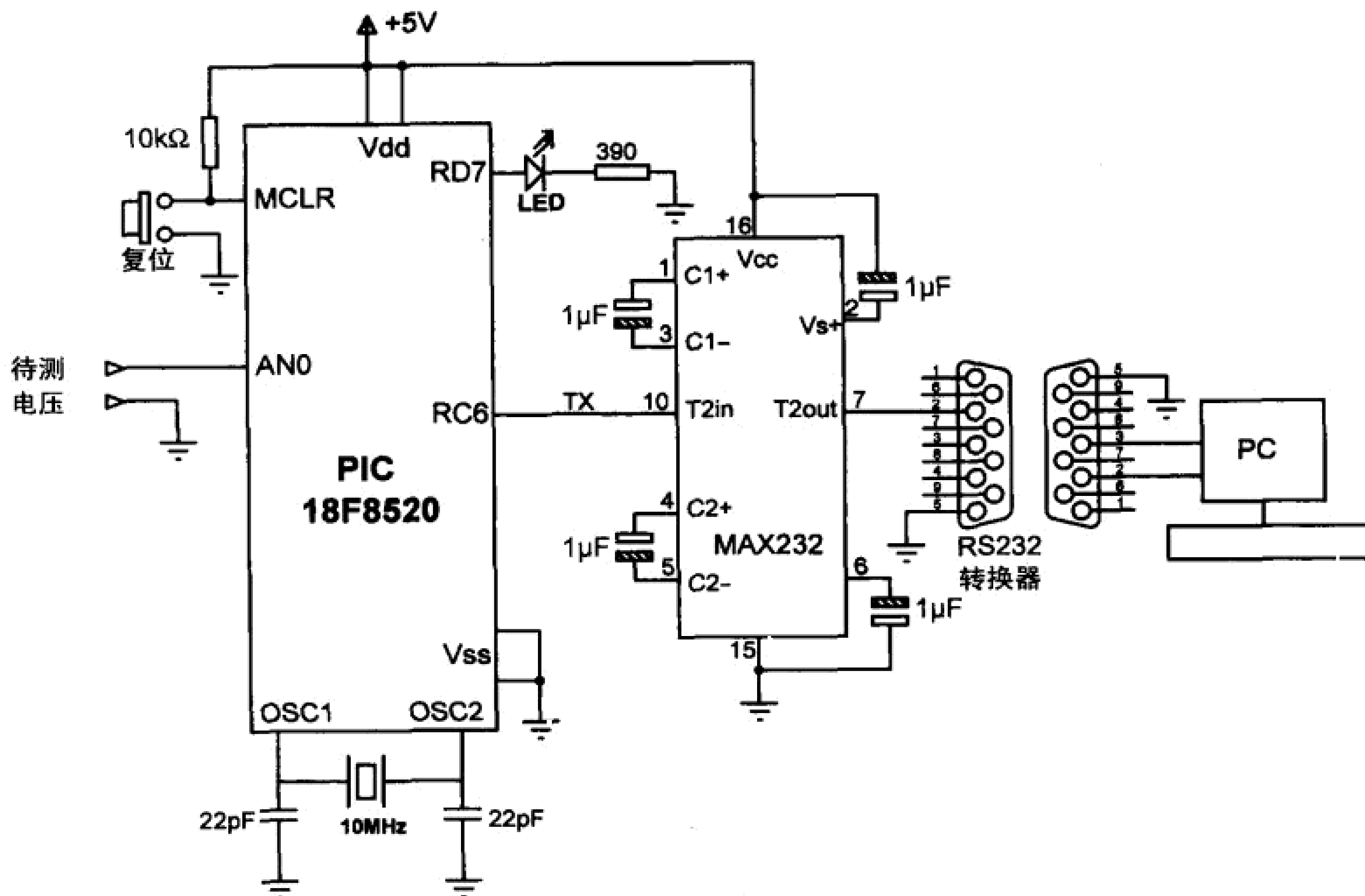


图10-13 项目的电路图

在PC上运行HyperTerminal程序，可以从程序中得到一个输出。图10-15给出了一个常见的屏幕输出。

```
/////////////////////////////////////////////////////////////////////////////////////
//
//          SIMPLE RTOS EXAMPLE - VOLTMETER WITH RS232 OUTPUT
//          ---------------------------------------------------------------
//
// This is a simple RTOS example. Analog voltage to be measured (between 0V
// and +5V) is connected to analog input AN0 of a PIC18F8520 type
// microcontroller. The microcontroller is operated from a 10MHz crystal. In
// addition, an LED is connected to port in RD7 of the microcontroller.
//
// RS232 serial output of the mirocontroller (RC6) is connected to a MAX232
// type RS232 voltage level converter chip. The output of this chip can be
// connected to the serial input of a PC (e.g., COM1) so that the measured
// voltage can be seen on the PC screen.
//
// The program consists of 3 tasks called "live", "Get_voltage", and "To_RS232".
//
// Task "Live" runs every 200ms and it flashes the LED conencted to port pin
// RD7 of the microcontroller to indicate that the program is running and is
// ready to measure voltages.
//
// task "Get_voltage" reads analog voltage from port AN0 and then converts
// the voltage into millivolts and stores in a variable called Volts.
```

图10-14 项目的程序清单

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
//
// Task "To_RS232" gets the measured voltage, converts it into a character
// array and then sends to the PC over the RS232 serial line. The serial line
// is configured to operate at 2400 Baud (higher Baud rates can also be used if
// desired).
//
// Programmer:        Dogan Ibrahim
// Date:              September, 2007
// File:              RTOS3.C
//
///////////////////////////////////////////////////////////////////////////////////

#include <18F8520.h>
#device adc=10
#use delay (clock=10000000)
#use rs232(baud=2400,xmit=PIN_C6,rcv=PIN_C7)

unsigned int16 adc_value;
unsigned int32 Volts;

//
// Define which timer to use and minor_cycle for RTOS
//
  #use rtos(timer=0, minor_cycle=100ms)

//
// Declare TASK "Live" - called every 200ms
// This task flashes the LED on port RD7
//
  #task(rate=200ms, max=1ms)
  void Live()
  {
    output_toggle(PIN_D7);                        // Toggle RD7 LED
  }


//
// Declare TASK "Get_voltage" - called every 10ms
//
  #task(rate=2s, max=100ms)
  void Get_voltage()
  {
    adc_value = read_adc();                       // Read A/D value
    Volts = (unsigned int32)adc_value*5000;
    Volts = Volts / 1024;                         // Voltage in mV
  }


//
// Declare TASK "To_RS232" - called every millisecond
//
  #task(rate=2s, max=100ms)
  void To_RS232()
  {
    printf("Measured Voltage = %LumV\n\r",Volts);   // send to RS232
  }



//
```

图10-14　（续）

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
// Start of MAIN program
//
void main()
{
   set_tris_d(0);                              // PORTD all outputs
   output_d(0);                                // Clear PORTD
   set_tris_a(0xFF);                           // PORTA all inputs
   setup_adc_ports(ALL_ANALOG);                // A/D ports
   setup_adc(ADC_CLOCK_DIV_32);                // A/D clock
   set_adc_channel(0);                         // Select channel 0 (AN0)
   delay_us(10);
   rtos_run();                                 // Start RTOS
}
```

图10-14 （续）

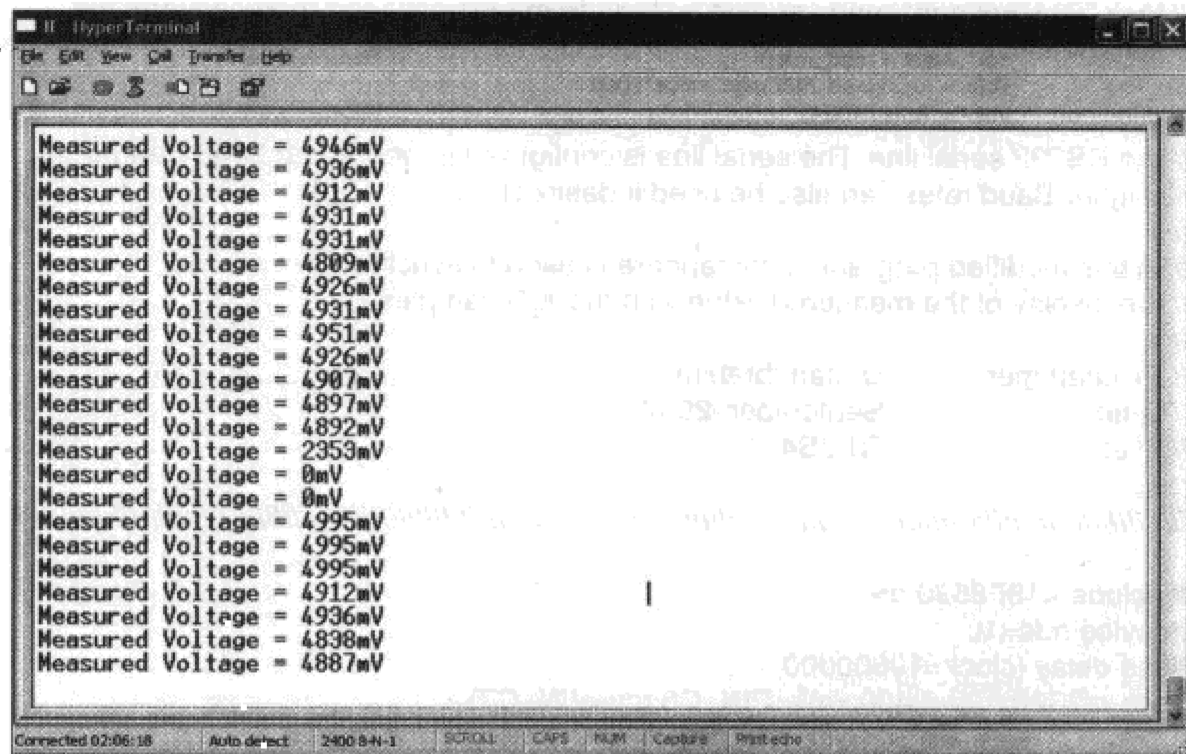


图10-15 程序的常见输出

使用一个信号量

图10-14给出的程序正在工作，并在PC屏幕上显示出所测量的电压值。使用信号量将该程序稍微修改一下，程序可以用来同步所测得电压值的显示和A/D采样值。图10-16给出了修改后的程序清单(RTOS4.C)。新程序的操作如下。

534～536

- 在程序开始时，将信号量变量（sem）设置为1。
- 任务Get_voltage用来递减信号量（调用函数rtos_wait）变量，以使任务To_RS232被阻断（信号量变量sem=0）并且不能向PC发送数据。当一个新的A/D采样值准备就绪时，信号量变量被递增（调用函数rtos_signal），任务To_RS232可以继续。于是，任务To_RS232将已测量到的电压值发送到PC，并增大信号量变量以表明成功读取数据。然后，任务Get_voltage用来得到一个新的采样值。这个过程将被永远地重复。

537～539

```
//////////////////////////////////////////////////////////////////////////////////
//
//        SIMPLE RTOS EXAMPLE - VOLTMETER WITH RS232 OUTPUT
//        ---------------------------------------------------------------
//
```

图10-16 修改后的程序清单

世纪电源网·论坛 BBS.21dianyuan.com 电源工程师 设计灵感之源


```
// This is a simple RTOS example. Analog voltage to be measured (between 0V
// and +5V) is connected to analog input AN0 of a PIC18F8520 type
// microcontroller. The microcontroller is operated from a 10MHz crystal. In
// addition, an LED is connected to port in RD7 of the microcontroller.
//
// RS232 serial output of the mirocontroller (RC6) is connected to a MAX232
// type RS232 voltage level converter chip. The output of this chip can be
// connected to the serial input of a PC (e.g., COM1) so that the measured
// voltage can be seen on the PC screen.
//
// The program consists of 3 tasks called "live", "Get_voltage", and "To_RS232".
//
// Task "Live" runs every 200ms and it flashes the LED connected to port RD7
// of the microcontroller to indicate that the program is running and is ready to
// measure voltages.
//
// task "Get_voltage" reads analog voltage from port AN0 and then converts the
// voltage into millivolts and stores in a variable called Volts.
//
// Task "To_RS232" gets the measured voltage  and then sends to the PC over
// the RS232 serial line. The serial line is configured to operate at 2400 Baud
// (higher Baud rates can also be used if desired).
//
// In this modified program, a semaphore is used to synchronize
// the display of the measured value with the A/D samples.
//
// Programmer:        Dogan Ibrahim
// Date:              September, 2007
// File:              RTOS4.C
//
//////////////////////////////////////////////////////////////////////////////////////

#include <18F8520.h>
#device adc=10
#use delay (clock=10000000)
#use rs232(baud=2400,xmit=PIN_C6,rcv=PIN_C7)

unsigned int16 adc_value;
unsigned int32 Volts;
int8 sem;

//
// Define which timer to use and minor_cycle for RTOS
//
  #use rtos(timer=0, minor_cycle=100ms)
//
// Declare TASK "Live" - called every 200ms
// This task flashes the LED on port RD7
//
  #task(rate=200ms, max=1ms)
  void Live()
  {
    output_toggle(PIN_D7);                          // Toggle RD7 LED
  }


//
// Declare TASK "Get_voltage" - called every 10ms
//
```

图10-16　（续）

PDG

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com


```
 #task(rate=2s, max=100ms)
 void Get_voltage()
 {
   rtos_wait(sem);                                   // decrement semaphore
   adc_value = read_adc();                           // Read A/D value
   Volts = (unsigned int32)adc_value*5000;
   Volts = Volts / 1024;                             // Voltage in mV
   rtos_signal(sem);                                 // increment semaphore
 }


//
// Declare TASK "To_RS232" - called every millisecond
//
 #task(rate=2s, max=100ms)
 void To_RS232()
 {
   rtos_wait(sem);                                   // Decrement semaphore
   printf("Measured Voltage = %LumV\n\r",Volts);     // Send to RS232
   rtos_signal(sem);                                 // Increment semaphore
 }


//
// Start of MAIN program
//
void main()
{
  set_tris_d(0);                                     // PORTD all outputs
  output_d(0);                                       // Clear PORTD
  set_tris_a(0xFF);                                  // PORTA all inputs
  setup_adc_ports(ALL_ANALOG);                       // A/D ports
  setup_adc(ADC_CLOCK_DIV_32);                       // A/D clock
  set_adc_channel(0);                                // Select channel 0 (AN0)

  delay_us(10);
  sem = 1;                                           // Semaphore is 1
  rtos_run();                                        // Start RTOS
}
```

图10-16 （续）

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

索　引

索引中的页码为英文原书页码，与本书页边标注的页码一致。

世纪电源网·论坛 电源工程师 BBS.21dianyuan.com 设计灵感之源

D

E

F

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

世纪电源网·论坛 电源工程师
BBS.21dianyuan.com 设计灵感之源



世纪电源网·论坛 电源工程师 设计灵感之源
BBS.21dianyuan.com

Advanced PIC Microcontroller Projects in C
From USB to RTOS with the PIC18F Series

tyw藏书

PIC项目实战

Microchip公司开发的PIC18系列微控制器，用于引脚数多、密度高的复杂应用。PIC18F微控制器提供性价比高的解决方案，用于使用RTOS和需要诸如TCP/IP、CAN、USB或者ZigBee这样复杂的通信协议栈，且用C语言实现通用应用系统。

本书基于PIC18F微控制器，深入介绍了使用microC语言设计PIC微控制器应用的方法。书中提供了C语言编程指导，microC编译器的使用贯穿始终，并有一章专门讲述microC函数和函数库。此外，本书还讨论了仿真器、模拟器和内电路调试器等开发工具，并举例说明了其在实际项目中的应用。

为便于读者学习掌握，书中给出了20多个PIC实际项目，包括：

- SD卡项目，如读CID寄存器并在PC屏幕上显示；
- 基于USB的项目，如基于USB的微控制器输入/输出端口；
- CAN总线项目，如温度传感器；
- RTOS项目，如随机数发生器。

Dogan Ibrahim 塞浦路斯近东大学计算机工程系主任，主要研究领域包括自动化控制、基于微处理器的设计、网络教育、远程教育和工程教育等。他写过40多本微处理器、微控制器及相关方面的书，并在各大技术期刊上发表过近200篇技术论文。



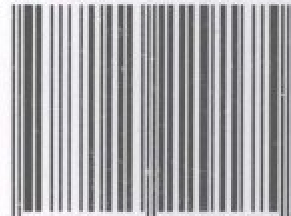
延伸阅读
- 嵌入式系统设计的艺术（第2版·英文版） 978-7-115-19521-0 49.00元
- PIC技术宝典 978-7-115-18554-9 99.00元
- PIC嵌入式系统开发 978-7-115-18265-4 69.00元
- 8051微控制器（第4版） 978-7-115-17959-3 49.00元
- 16位单片机C语言编程：基于PIC24 978-7-115-22149-0 49.00元
- 32位单片机C语言编程：基于PIC32 978-7-115-21612-0 49.00元

本书译自原版*Advanced PIC Microcontroller Projects in C: From USB to RTOS with the PIC18F Series*，并由Elsevier授权出版。









图灵网站 http://www.turingbook.com 热线：(010) 51095186
反馈/投稿/推荐信箱：contact@turingbook.com
有奖勘误：debug@turingbook.com

分类建议：电子电气／单片机
人民邮电出版社网址 www.ptpress.com.cn

ISBN 978-7-115-22917-5
定价：55.00元